



小雨淅淅o0

[博客园](#) [首页](#) [新随笔](#) [联系](#) [订阅](#) [管理](#)

Nginx调优

前言：

最近相对比较闲，整理下以前工作中学到的东西，将接触到生产环境中的内容都深入学习一下，nginx与tomcat的优化是在山西ott项目上接触到的，直接操作配置文件很容易，但更多的配置参数与优化方式还是需要了解的，先得知道问题在哪里，然后再做。

Nginx调优

公告

昵称： 小雨淅淅o0
园龄： 4年3个月
粉丝： 40
关注： 18
[+加关注](#)

<	2023年5月						>
日	一	二	三	四	五	六	
30	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
--	--	--	--	-	-	-	

目录

软件调优

- 1.隐藏 Nginx 版本号
 - 2.隐藏 Nginx 版本号和软件名
 - 3.更改 Nginx 服务的默认用户
 - 4.优化 Nginx worker 进程数
 - 5.绑定 Nginx 进程到不同的 CPU 上
 - 6.优化 Nginx 处理事件模型
 - 7.优化 Nginx 单个进程允许的最大连接数
 - 8.优化 Nginx worker 进程最大打开文件数
 - 9.优化服务器域名的散列表大小
 - 10.开启高效文件传输模式
 - 11.优化 Nginx 连接超时时间
 - 12.限制上传文件的大小
 - 13.FastCGI 相关参数调优
 - 14.配置 Nginx gzip 压缩
 - 15.配置 Nginx expires 缓存
 - 16.优化 Nginx日志(日志切割)
 - 17.优化 Nginx 站点目录
 - 18.配置 Nginx 防盗链
 - 19.配置 Nginx 错误页面优雅显示
 - 20.优化 Nginx 文件权限
 - 21.Nginx 防爬虫优化
 - 22.控制 Nginx 并发连接数
 - 23. 集群代理优化
- 系统内核参数优化

Nginx软件调优

1. 隐藏 Nginx 版本号

28	29	30	31	1	2	3
4	5	6	7	8	9	10

搜索

找找看

谷歌搜索

随笔分类

ansible(4)

CI/CD(6)

Docker(12)

EFK(8)

Interview preparation(1)

IT基础问题与实用小技巧(22)

kubernetes knowledge(36)

kubernetes问题处理(8)

mongo(4)

为什么要隐藏 Nginx 版本号：一般来说，软件的漏洞都与版本有关，隐藏版本号是为了防止恶意用户利用软件漏洞进行攻击

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2     worker_processes  1;
3     events {
4         worker_connections  1024;
5     }
6     http {
7         include        mime.types;
8         default_type    application/octet-stream;
9         sendfile        on;
10        keepalive_timeout  65;
11        server_tokens    off;      # 隐藏版本号
12        server {
13            listen        80;
14            server_name    www.abc.com;
15            location / {
16                root       html/www;
17                index       index.html index.htm;
18            }
19        }
20    }
21    ...
22
23 [root@localhost ~]# /usr/local/nginx/sbin/nginx -t
24 [root@localhost ~]# /usr/local/nginx/sbin/nginx -s reload
25
26 [root@localhost ~]# curl -I 127.0.0.1      # 查看是否隐藏版本号
27 HTTP/1.1 404 Not Found
28 Server: nginx
29 Date: Thu, 25 May 2017 05:23:03 GMT
30 Content-Type: text/html
31 Content-Length: 162
32 Connection: keep-alive
```

mysql(22)

nginx、tomcat、apache(10)

oracle(4)

prometheus(5)

python(11)

shell(17)

更多

随笔档案

2023年3月(1)

2022年9月(1)

2022年6月(1)

2022年5月(3)

2022年4月(1)

2.隐藏 Nginx 版本号和软件名

为什么要隐藏 Nginx 版本号和软件名：一般来说，软件的漏洞都与版本有关，隐藏版本号是为了防止恶意用户利用软件漏洞进行攻击，而软件名可以进行修改，否则黑客知道是 Nginx 服务器更容易进行攻击，需要注意的是，隐藏 Nginx 软件名需要重新编译安装 Nginx，如果没有该方面需求尽量不要做

1) 修改：/usr/local/src/nginx-1.6.3/src/core/nginx.h

```
1 | #define NGINX_VERSION      "8.8.8.8"           # 修改为想要显示的版本号
2 | #define NGINX_VER         "Google/" NGINX_VERSION # 修改为想要显示的软件名
3 | #define NGINX_VAR         "Google"             # 修改为想要显示的软件名
```

2) 修改：/usr/local/src/nginx-1.6.3/src/http/ngx_http_header_filter_module.c

```
1 | static char ngx_http_server_string[] = "Server: Google" CRLF;      # 修改为想要显示的软件名
2 | static char ngx_http_server_full_string[] = "Server: " NGINX_VER CRLF;
```

3) 修改：/usr/local/src/nginx-1.6.3/src/http/ngx_http_special_response.c

```
1 | static u_char ngx_http_error_full_tail[] =
2 | "<hr><center>" NGINX_VER "(www.google.com)</center>" CRLF # 此行定义对外展示的内容
3 | "</body>" CRLF
4 | "</html>" CRLF
5 | ;
6 |
7 | static u_char ngx_http_error_tail[] =
8 | "<hr><center>Google</center>" CRLF # 此行定义对外展示的软件名
9 | "</body>" CRLF
10 | "</html>" CRLF
11 | ;
```

4) 重新编译 Nginx

```
1 | cd /usr/local/src/nginx-1.6.3
2 | ./configure --user=nginx --group=nginx --prefix=/usr/local/nginx --with-http_stub_status_module --with-http_ssl_module
3 | make && make install
4 | /usr/local/nginx/sbin/nginx
```

2022年2月(4)

2021年11月(5)

2021年10月(21)

2021年9月(1)

2021年8月(1)

2021年6月(2)

2021年5月(4)

2021年4月(1)

2021年3月(1)

2021年1月(1)

更多

阅读排行榜

1. hyper-v简介及安装使用(34370)

2. prometheus和zabbix的对比(32009)

3. shell中数组的定义与操作(19979)

4. docker-compose简介及使用(17500)

5. 使用docker快速安装oracle(17131)

评论排行榜

1. 使用docker快速安装oracle(2)

2. k8s证书过期的处理方法(2)

3. k8s的二进制包安装(2)

4. zabbix通过snmp监控windows主机(2)

5. 如何转发博客园中的文章(2)

推荐排行榜

1. K8s网络插件flannel与calico(5)

2. prometheus和zabbix的对比(5)

3.更改 Nginx 服务的默认用户

为什么要更改 Nginx 服务的默认用户：就像更改 ssh 的默认 22 端口一样，增加安全性，Nginx 服务的默认用户是 nobody，我们更改为 nginx

1) 添加 nginx 用户

```
1 useradd -s/sbin/nologin -M nginx
```

2) 更改 Nginx 配置文件

```
1 [root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf
2 worker_processes 1;
3 user nginx nginx;           # 指定Nginx服务的用户和用户组
4 events {
5     worker_connections 1024;
6 }
7 http {
8     include mime.types;
9     default_type application/octet-stream;
10    sendfile on;
11    keepalive_timeout 65;
12    server_tokens off;
13    server {
14        listen 80;
15        server_name www.abc.com;
16        location / {
17            root html/www;
18            index index.html index.htm;
19        }
20    }
21 }
```

3) 重新加载 Nginx

```
1 [root@localhost ~]# /usr/local/nginx/sbin/nginx -t
2 [root@localhost ~]# /usr/local/nginx/sbin/nginx -s reload
```

4) 验证是否生效

```
1 [root@localhost ~]# ps aux | grep nginx
2 root      8901  0.0  0.1 45036 1784 ?        Ss   13:54   0:00 nginx: master process /usr/local/nginx/sbin/nginx
3 nginx     8909  0.0  0.1 45460 1828 ?        S    13:59   0:00 nginx: worker process # Nginx进程的所属用户为r
```

4.优化 Nginx worker 进程数

Nginx 有 Master 和 worker 两种进程，Master 进程用于管理 worker 进程，worker 进程用于 Nginx 服务

worker 进程数应该设置为等于 CPU 的核数，高流量并发场合也可以考虑将进程数提高至 CPU 核数 * 2

```
1 [root@localhost ~]# grep -c processor /proc/cpuinfo # 查看CPU核数
2 2
3
4 [root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf # 设置worker进程数
5 worker_processes 2;
6 user nginx nginx;
7 .....
8
9 [root@localhost ~]# /usr/local/nginx/sbin/nginx -t # 重新加载Nginx
10 [root@localhost ~]# /usr/local/nginx/sbin/nginx -s reload
11
12 [root@localhost ~]# ps -ef | grep nginx | grep -v grep # 验证是否为设置的进程数
13 root      8901      1  0 13:54 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
14 nginx     8937    8901  0 14:14 ?        00:00:00 nginx: worker process
15 nginx     8938    8901  0 14:14 ?        00:00:00 nginx: worker process
```

5.绑定 Nginx 进程到不同的 CPU 上

为什么要绑定 Nginx 进程到不同的 CPU 上：默认情况下，Nginx 的多个进程有可能跑在某一个 CPU 或 CPU 的某一核上，导致 Nginx 进程使用硬件的资源不均，因此绑定 Nginx 进程到不同的 CPU 上是为了充分利用硬件的多 CPU 多核资源的目的。

```
1 [root@localhost ~]# grep -c processor /proc/cpuinfo # 查看CPU核数
2 2
3 worker_processes 2; # 2核CPU的配置
```

3. 使用docker快速安装oracle(4)

4. 如何转发博客园中的文章(3)

5. linux病毒扫描工具ClamAV使用(2)

最新评论

1. Re:zabbix通过snmp监控windows主机

@白色温哥华 如果是测试环境的话，检查windows配置，如果是生产环境，多考虑防火墙的问题，需要用nc命令测试snmp端口，zabbix端口等...

--小雨浙浙o0

2. Re:zabbix通过snmp监控windows主机

所以问题都检查过了，就是一直为灰色

--白色温哥华

3. Re:使用docker快速安装oracle

已经试了m1 版本的不支持

--coderSisyphus

4. Re:K8s网络插件flannel与calico

```

4 worker_cpu_affinity01 10;
5
6 worker_processes 4;          # 4核CPU的配置
7 worker_cpu_affinity0001 0010 0100 1000;
8
9 worker_processes 8;          # 8核CPU的配置
10 worker_cpu_affinity00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;
11
12 [root@localhost ~]# /usr/local/nginx/sbin/nginx -t
13 [root@localhost ~]# /usr/local/nginx/sbin/nginx -s reload

1 [root@localhost ~]# cd /usr/local/src/ # 进行压力测试, 教程: http://os.51cto.com/art/201202/317803.htm
2 [root@localhost src]# wget http://home.tiscali.cz/~cz210552/distfiles/webbench-1.5.tar.gz
3 [root@localhost src]# tar -zxvf webbench-1.5.tar.gz
4 [root@localhost src]# cd webbench-1.5
5 [root@localhost src]# yum install -y ctags gcc
6 [root@localhost src]# mkdir -m 644 -p /usr/local/man/man1
7 [root@localhost src]# make && make install
8 [root@localhost src]# webbench -c 10000 -t 60 http://192.168.5.131/

1 [root@localhost ~]# top # 按1查看CPU调度结果, 这里是虚拟机测试, 效果并不太明显
2 top - 14:44:46 up 4:40, 3 users, load average: 0.01, 0.32, 0.24
3 Tasks: 85 total, 1 running, 84 sleeping, 0 stopped, 0 zombie
4 Cpu0 : 0.8%us, 0.8%sy, 0.0%ni, 97.9%id, 0.3%wa, 0.0%hi, 0.2%si, 0.0%st
5 Cpu1 : 0.6%us, 0.7%sy, 0.0%ni, 98.1%id, 0.0%wa, 0.0%hi, 0.5%si, 0.0%st
6 Mem: 1534840k total, 304824k used, 1230016k free, 3932k buffers
7 Swap: 204792k total, 0k used, 204792k free, 191364k cached
8
9 PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
10 4319 root 20 0 98308 3932 2964 S 3.2 0.3 0:15.76 sshd
11 18989 root 20 0 15016 1292 1008 R 1.6 0.1 0:00.04 top
12 1 root 20 0 19232 1388 1112 S 0.0 0.1 0:02.19 init
13 2 root 20 0 0 0 0 S 0.0 0.0 0:00.08 kthreadd
14 3 root RT 0 0 0 0 S 0.0 0.0 0:00.61 migration/0
15 4 root 20 0 0 0 0 S 0.0 0.0 0:03.60 ksoftirqd/0
16 5 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
17 6 root RT 0 0 0 0 S 0.0 0.0 0:00.50 watchdog/0

```

mark 一下

--寒@鹏

5. Re:使用docker快速安装oracle

m1支持? ? ?

试了几个版本都不行

--TinaRoot

6.优化 Nginx 处理事件模型

Nginx 的连接处理机制在不同的操作系统会采用不同的 I/O 模型，要根据不同的系统选择不同的事件处理模型，可供选择的事件处理模型有：kqueue、rtsig、epoll、/dev/poll、select、poll，其中 select 和 epoll 都是标准的工作模型，kqueue 和 epoll 是高效的工作模型，不同的是 epoll 用在 Linux 平台上，而 kqueue 用在 BSD 系统中。

- (1) 在 Linux 下，Nginx 使用 epoll 的 I/O 多路复用模型
- (2) 在 Freebsd 下，Nginx 使用 kqueue 的 I/O 多路复用模型
- (3) 在 Solaris 下，Nginx 使用 /dev/poll 方式的 I/O 多路复用模型
- (4) 在 Windows 下，Nginx 使用 icop 的 I/O 多路复用模型

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 .....
3 events {
4     use epoll;
5 }
6 .....
```

7.优化 Nginx 单个进程允许的最大连接数

- (1) 控制 Nginx 单个进程允许的最大连接数的参数为 worker_connections，这个参数要根据服务器性能和内存使用量来调整
- (2) 进程的最大连接数受 Linux 系统进程的最大打开文件数限制，只有执行了 "ulimit -HSn 65535" 之后，worker_connections 才能生效
- (3) 连接数包括代理服务器的连接、客户端的连接等，Nginx 总并发连接数 = worker 数量 * worker_connections, 总数保持在3w左右

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 worker_processes 2;
3 worker_cpu_affinity01 10;
4 user nginx nginx;
5 events {
6     use epoll;
7     worker_connections 15000;
8 }
9 .....
```


8.优化 Nginx worker 进程最大打开文件数

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 worker_processes 2;
3 worker_cpu_affinity 01 10;
4 worker_rlimit_nofile 65535;    # worker 进程最大打开文件数，可设置为优化后的 ulimit -HSn 的结果
5 user nginx nginx;
6 events {
7     worker_connections 1024;
8 }
9 http {
10     include mime.types;
11     default_type application/octet-stream;
12     sendfile on;
13     keepalive_timeout 65;
14     server_tokens off;
15     server {
16         listen 80;
17         server_name www.abc.com;
18         location / {
19             root html/www;
20             index index.html index.htm;
21         }
22     }
23 }
```

9.优化服务器域名的散列表大小

如下，如果在 server_name 中配置了一个很长的域名，那么重载 Nginx 时会报错，因此需要使用 server_names_hash_max_size 来解决域名过长的问题，该参数的作用是设置存放域名的最大散列表的存储的大小，根据 CPU 的一级缓存大小来设置。

```
1 server {
2     listen 80;
3     server_name www.abcdefghijklmnpqrst.com;    # 配置一个很长的域名
4     location / {
5         root html/www;
```

```
6         index index.html index.htm;
7     }
8 }

1 [root@localhost conf]# /usr/local/nginx/sbin/nginx -t    # 如果配置的域名很长会出现如下错误
2 nginx: [emerg] could not build the server_names_hash, you should increase server_names_hash_bucket_size:64
3 nginx: configuration file /usr/local/nginx/conf/nginx.conf test failed

1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 .....
3 http {
4     include      mime.types;
5     server_names_hash_bucket_size 512;    # 配置在 http 区块, 默认是 512kb , 一般设置为 cpu 一级缓存的 4-5 倍, 一级缓存
6     default_type application/octet-stream;
7     sendfile      on;
8     keepalive_timeout 65;
9     server_tokens off;
10    include vhosts/*.conf;
11 }
```

10. 开启高效文件传输模式

(1) sendfile 参数用于开启文件的高效传输模式, 该参数实际上是激活了 sendfile() 功能, sendfile() 是作用于两个文件描述符之间的数据拷贝函数, 这个拷贝操作是在内核之中的, 被称为 "零拷贝", sendfile() 比 read 和 write 函数要高效得多, 因为 read 和 write 函数要把数据拷贝到应用层再进行操作

(2) tcp_nopush 参数用于激活 Linux 上的 TCP_CORK socket 选项, 此选项仅仅当开启 sendfile 时才生效, tcp_nopush 参数可以允许把 http response header 和文件的开始部分放在一个文件里发布, 以减少网络报文段的数量

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 .....
3 http {
4     include      mime.types;
5     server_names_hash_bucket_size 512;
6     default_type application/octet-stream;
7     sendfile      on;    # 开启文件的高效传输模式
```

```
8      tcp_nopush    on;    # 激活 TCP_CORK socket 选择
9      tcp_nodelay on; #数据在传输的过程中不进缓存
10     keepalive_timeout 65;
11     server_tokens off;
12     include vhosts/*.conf;
13 }
```

11.优化 Nginx 连接超时时间

1. 什么是连接超时

- (1) 举个例子，某饭店请了服务员招待顾客，但是现在饭店不景气，因此要解雇掉一些服务员，这里的服务员就相当于 Nginx 服务建立的连接
- (2) 当服务器建立的连接没有接收处理请求时，可以在指定的时间内让它超时自动退出

2. 连接超时的作用

- (1) 将无用的连接设置为尽快超时，可以保护服务器的系统资源（CPU、内存、磁盘）
- (2) 当连接很多时，及时断掉那些建立好的但又长时间不做事的连接，以减少其占用的服务器资源
- (3) 如果黑客攻击，会不断地和服务器建立连接，因此设置连接超时以防止大量消耗服务器的资源
- (4) 如果用户请求了动态服务，则 Nginx 就会建立连接，请求 FastCGI 服务以及后端 MySQL 服务，设置连接超时，使得在用户容忍的时间内返回数据

3. 连接超时存在的问题

- (1) 服务器建立新连接是要消耗资源的，因此，连接超时时间不宜设置得太短，否则会造成并发很大，导致服务器瞬间无法响应用户的请求
- (2) 有些 PHP 站点会希望设置成短连接，因为 PHP 程序建立连接消耗的资源和时间相对要少些
- (3) 有些 Java 站点会希望设置成长连接，因为 Java 程序建立连接消耗的资源和时间要多一些，这时由语言的运行机制决定的

4. 设置连接超时

- (1) `keepalive_timeout` ：该参数用于设置客户端连接保持会话的超时时间，超过这个时间服务器会关闭该连接
- (2) `client_header_timeout` ：该参数用于设置读取客户端请求头数据的超时时间，如果超时客户端还没有发送完整的 header 数据，服务器将返回 "Request time out (408)" 错误
- (3) `client_body_timeout` ：该参数用于设置读取客户端请求主体数据的超时时间，如果超时客户端还没有发送完整的主体数据，服务器将返回 "Request time out (408)" 错误

- (4) `send_timeout` : 用于指定响应客户端的超时时间, 如果超过这个时间, 客户端没有任何活动, Nginx 将会关闭连接
- (5) `tcp_nodelay` : 默认情况下当数据发送时, 内核并不会马上发送, 可能会等待更多的字节组成一个数据包, 这样可以提高 I/O 性能, 但是, 在每次只发送很少字节的业务场景中, 使用 `tcp_nodelay` 功能, 等待时间会比较长

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 .....
3 http {
4     include      mime.types;
5     server_names_hash_bucket_size 512;
6     default_type  application/octet-stream;
7     sendfile      on;
8     keepalive_timeout 65;
9     tcp_nodelay on;
10    client_header_timeout15;
11    client_body_timeout15;
12    send_timeout25;
13    include vhosts/*.conf;
14 }
```

12.限制上传文件的大小

`client_max_body_size` 用于设置最大的允许客户端请求主体的大小, 在请求首部中有 "Content-Length", 如果超过了此配置项, 客户端会收到 413 错误, 即请求的条目过大

```
1 worker_processes 2;
2 worker_cpu_affinity 01 10;
3 user nginx nginx;
4 error_log logs/error.log error;
5
6 events {
7     use epoll;
8     worker_connections 20480;
9 }
10
11 http {
12     include      mime.types;
```

```
13     server_names_hash_bucket_size 512;
14     default_type  application/octet-stream;
15     sendfile        on;
16     keepalive_timeout 65;
17     server_tokens off;
18     client_max_body_size 8m;    # 设置客户端最大的请求主体大小为8M
19     include vhosts/*.conf;
20 }
```

13.FastCGI 相关参数调优

当 LNMP 组合工作时，首先是用户通过浏览器输入域名请求 Nginx Web 服务，如果请求的是静态资源，则由 Nginx 解析返回给用户；如果是动态请求（如 PHP），那么 Nginx 就会把它通过 FastCGI 接口发送给 PHP 引擎服务（即 php-fpm）进行解析，如果这个动态请求要读取数据库数据，那么 PHP 就会继续向后请求 MySQL 数据库，以读取需要的数据，并最终通过 Nginx 服务把获取的数据返回给用户，这就是 LNMP 环境的基本请求流程。

FastCGI 介绍：CGI 通用网关接口，是 HTTP 服务器与其他机器上的程序服务通信交流的一种工具，CGI 接口的性能较差，每次 HTTP 服务器遇到动态程序时都需要重新启动解析器来执行解析，之后结果才会被返回 HTTP 服务器，因此就有了 FastCGI，FastCGI 是一个在 HTTP 服务器和动态脚本语言间通信的接口，主要是把动态语言和 HTTP 服务器分离开来，使得 HTTP 服务器专一地处理静态请求，提高整体性能，在 Linux 下，FastCGI 接口即为 socket，这个 socket 可以是文件 socket 也可以是 IP socket

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 worker_processes 1;
3 events {
4     worker_connections 1024;
5 }
6 http {
7     include        mime.types;
8     default_type  application/octet-stream;
9     sendfile        on;
10    keepalive_timeout 65;
11    fastcgi_connect_timeout 240;    # Nginx服务器和后端FastCGI服务器连接的超时时间
12    fastcgi_send_timeout    240;    # Nginx允许FastCGI服务器返回数据的超时时间，即在规定时间内后端服务器必须传完所有的数据
13    fastcgi_read_timeout    240;    # Nginx从FastCGI服务器读取响应信息的超时时间，表示连接建立成功后，Nginx等待后端服务器
14    fastcgi_buffer_size     64k;    # Nginx FastCGI 的缓冲区大小，用来读取从FastCGI服务器端收到的第一部分响应信息的缓冲区
```

```
15 fastcgi_buffers      4 64k;    # 设定用来读取从FastCGI服务器端收到的响应信息的缓冲区大小和缓冲区数量
16 fastcgi_busy_buffers_size 128k; # 用于设置系统很忙时可以使用的 proxy_buffers 大小
17 fastcgi_temp_file_write_size128k; # FastCGI 临时文件的大小
18 # fastcti_temp_path    /data/nginx_fcgi_tmp;    # FastCGI 临时文件的存放路径
19 fastcgi_cache_path    /data/nginx_fcgi_cache levels=2:2 keys_zone=ngx_fcgi_cache:512m inactive=1d max
20
21 server {
22     listen      80;
23     server_name www.abc.com;
24     location/ {
25         root    html/www;
26         index   index.html index.htm;
27     }
28     location ~ .*\. (php|php5)?$ {
29         root            html/www;
30         fastcgi_pass     127.0.0.1:9000;
31         fastcgi_index    index.php;
32         include          fastcgi.conf;
33         fastcgi_cache    ngx_fcgi_cache;          # 缓存FastCGI生成的内容, 比如PHP生成的动态内容
34         fastcgi_cache_valid 200 302 1h;          # 指定http状态码的缓存时间, 这里表示将200和302缓存1小时
35         fastcgi_cache_valid 301 1d;              # 指定http状态码的缓存时间, 这里表示将301缓存1天
36         fastcgi_cache_valid any 1m;              # 指定http状态码的缓存时间, 这里表示将其他状态码缓存1分钟
37         fastcgi_cache_min_uses 1;                # 设置请求几次之后响应被缓存, 1表示一次即被缓存
38         fastcgi_cache_use_stale error timeout invalid_header http_500; # 定义在哪些情况下使用过期缓存
39         fastcgi_cache_key   http://$host$request_uri; # 定义 fastcgi_cache 的 key
40     }
41 }
42 }
```

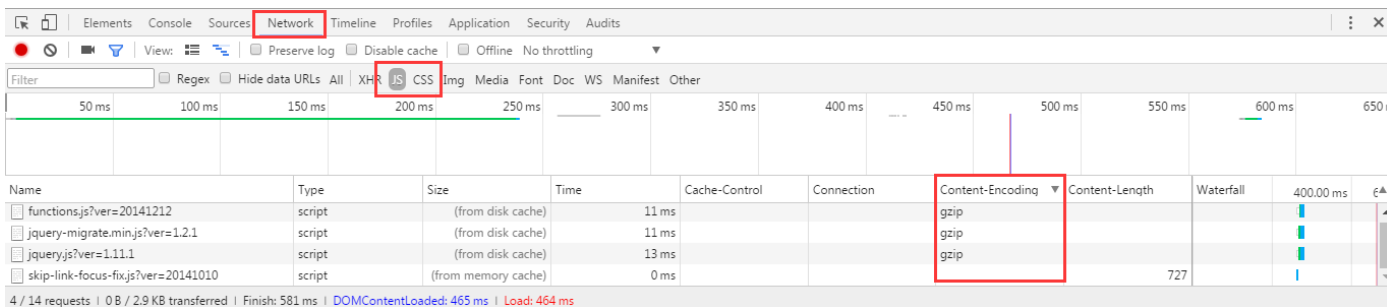
```
1 [root@localhost ~]# pkill php-fpm
2 [root@localhost ~]# /usr/local/php/sbin/php-fpm
```

14.配置 Nginx gzip 压缩

Nginx gzip 压缩模块提供了压缩文件内容的功能，用户请求的内容在发送到客户端之前，Nginx 服务器会根据一些具体的策略实施压缩，以节约网站出口带宽，同时加快数据传输效率，来提升用户访问体验，需要压缩的对象有 html、js、css、xml、shtml，图片和视频尽量不要压缩，因为这些文件大多都是已经压缩过的，如果再压缩可能反而变大，另外，压缩的对象必须大于 1KB，由于压缩算法的特殊原因，极小的文件压缩后可能反而变大

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 .....
3 http {
4     gzip on;                # 开启压缩功能
5     gzip_min_length 1k;     # 允许压缩的对象的最小字节
6     gzip_buffers 4 32k;     # 压缩缓冲区大小，表示申请4个单位为16k的内存作为压缩结果的缓存
7     gzip_http_version 1.1;  # 压缩版本，用于设置识别HTTP协议版本
8     gzip_comp_level 9;      # 压缩级别，1级压缩比最小但处理速度最快，9级压缩比最高但处理速度最慢
9     gzip_types text/css text/xml application/javascript; # 允许压缩的媒体类型
10    gzip_vary on;           # 该选项可以让前端的缓存服务器缓存经过gzip压缩的页面，例如用代理服务器缓存经过Nginx压缩的
11 }
```

检查压缩：可以用 Google 浏览器按 F12 查看，也可以在 Google 浏览器安装 yslow 插件 (yslow.org)



The screenshot shows the Network tab in Google Chrome DevTools. The 'Content-Encoding' column for several resources is highlighted with a red box, showing 'gzip'. The 'Content-Length' column for the same resources is empty, indicating successful compression. The 'Waterfall' column shows the timing of each resource load.

Name	Type	Size	Time	Cache-Control	Connection	Content-Encoding	Content-Length	Waterfall
functions.js?ver=20141212	script	(from disk cache)	11 ms			gzip		
jquery-migrate.min.js?ver=1.2.1	script	(from disk cache)	11 ms			gzip		
jquery.js?ver=1.11.1	script	(from disk cache)	13 ms			gzip		
skip-link-focus-fix.js?ver=20141010	script	(from memory cache)	0 ms				727	

4 / 14 requests | 0 B / 2.9 KB transferred | Finish: 581 ms | DOMContentLoaded: 465 ms | Load: 464 ms

chrome-extension://ninejcohidippngpapiilnmkgllmakh/yslow.html#1

Home Grade Components Statistics

Components The page has a total of **10** components and a total weight of **257.2K** bytes

TYPE	SIZE (KB)	↓ GZIP (KB)	COOKIE RECEIVED (bytes)	COOKIE SENT (bytes)	HEADERS	URL
js	95.8K	95.8K			🔍	http://www.wordpress.com/wp-includes/
css	92.3K	92.3K			🔍	http://www.wordpress.com/wp-content/
css	27.5K	27.5K			🔍	http://www.wordpress.com/wp-content/themes/twentyfifteen/genericon
css	22.6K	22.6K			🔍	http://fonts.googleapis.com/css?...
js	7.2K	7.2K			🔍	http://www.wordpress.com/wp-includes/

15.配置 Nginx expires 缓存

- (1) Nginx expires 的功能就是为用户访问的网站内容设定一个过期时间，当用户第一次访问这些内容时，会把这些内容存储在用户浏览器本地，这样用户第二次及以后继续访问该网站时，浏览器会检查加载已经缓存在用户浏览器本地的内容，就不会去服务器下载了，直到缓存的内容过期或被清除为止
- (2) 不希望被缓存的内容：广告图片、网站流量统计工具、更新很频繁的文件
- (3) 缓存日期参考：51CTO 缓存 1 周，新浪缓存 15 天，京东缓存 25 年，淘宝缓存 10 年

```

1  server {
2      listen      80;
3      server_name www.abc.com abc.com;
4      root        html/www;
5      location ~ .*\. (gif|jpg|jpeg|png|bmp|swf|js|css)$ # 缓存的对象
6      {
7          expires  3650d; # 缓存的时间，3650天，即10年
8      }
9  }
```

使用 Google 浏览器安装 yslow 插件来查看：

chrome-extension://ninejcohidippngpapiilmkgllmakh/yslow.html#1

Home Grade Components Statistics Rulesets YSlow(V2) Edit Help

Components The page has a total of 12 components and a total weight of 263.1K bytes

TYPE	SIZE (KB)	GZIP (KB)	COOKIE RECEIVED (bytes)	COOKIE SENT (bytes)	HEADERS	URL	EXPIRES (Y/M/D)	RESPONSE TIME (ms)	ETAG	ACTION
js	0.7K				⌘	http://www.wordpress.com/wp-includes/js/comment-reply.min.js?...	2027/5/5	0	"5285a621-2f5"	
image	1.2K				⌘	http://0.gravatar.com/avatar/ad516503a11cd5ca435acc9bb6523536?...	2017/5/7	0		smush.it
css	22.6K	22.6K			⌘	http://fonts.googleapis.com/css?...	2017/5/7	0		
doc	10.8K	10.8K			⌘	http://www.wordpress.com/archives/1.html	no	0		

16.优化 Nginx access 日志

1. 配置日志切割

```

1 [root@localhost ~]# vim /usr/local/nginx/conf/cut_nginx_log.sh
2 #!/bin/bash
3 savepath_log='/usr/local/clogs'
4 nglogs='/usr/local/nginx/logs'
5 mkdir-p $savepath_log/$(date+%Y)/$(date+%m)
6 mv $nglogs/access.log $savepath_log/$(date+%Y)/$(date+%m)/access.$(date+%Y%m%d).log
7 mv $nglogs/error.log $savepath_log/$(date+%Y)/$(date+%m)/error.$(date+%Y%m%d).log
8 kill-USR1 `cat/usr/local/nginx/logs/nginx.pid`

1 [root@localhost ~]# crontab -e # 每天凌晨0点执行脚本
2 0 0 * * * /bin/sh/usr/local/nginx/conf/cut_nginx_log.sh >/dev/null2>&1

```

2. 不记录不需要的访问日志

```

1 location ~ .*\. (js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF)$ {
2     access_log off;
3 }

```

3. 设置访问日志的权限

```

1 chown-R root.root/usr/local/nginx/logs
2 chmod-R700 /usr/local/nginx/logs

```

17.优化 Nginx 站点目录

1. 禁止解析指定目录下的指定程序

```

1 location ~ ^/data/.*\.(php|php5|sh|pl|py)$ { # 根据实际来禁止哪些目录下的程序, 且该配置必须写在 Nginx 解析 PHP 的配置前
2     denyall;
3 }

```

2. 禁止访问指定目录

```

1 location ~ ^/data/.*\.(php|php5|sh|pl|py)$ { # 根据实际来禁止哪些目录下的程序, 且该配置必须写在 Nginx 解析 PHP 的配置前
2     denyall;
3 }

```

3. 限制哪些 IP 不能访问网站

```

1 location ~ ^/wordpress { # 相对目录, 表示只允许 192.168.1.1 访问网站根目录下的 wordpress 目录
2     allow192.168.1.1/24;
3     denyall;
4 }

```

18.配置 Nginx 防盗链

什么是防盗链: 简单地说, 就是某些不法网站未经许可, 通过在其自身网站程序里非法调用其他网站的资源, 然后在自己的网站上显示这些调用的资源, 使得被盗链的那一端消耗带宽资源 (1) 根据 HTTP referer 实现防盗链: referer 是 HTTP的一个首部字段, 用于指明用户请求的 URL 是从哪个页面通过链接跳转过来的

(2) 根据 cookie 实现防盗链: cookie 是服务器贴在客户端身上的 "标签", 服务器用它来识别客户端

根据 referer 配置防盗链:

```

1 #第一种,匹配后缀
2 location ~ .*\. (gif|jpg|jpeg|png|bmp|swf|flv|rar|zip|gz|bz2)$ { # 指定需要使用防盗链的媒体资源
3     access_log off; # 不记录防盗链的日志
4     expires 15d; # 设置缓存时间
5     valid_referers none blocked *.test.com *.abc.com; # 表示这些地址可以访问上面的媒体资源
6     if ($invalid_referer) { # 如果地址不如上面指定的地址就返回403
7         return 403
8     }
9 }

```

```
10
11 #第二种,绑定目录
12 location /images {
13     root /web/www/img;
14     vaild_referers nono blocked *.spdir.com *.spdir.top;
15     if ($invalid_referer) {
16         return 403;
17     }
18 }
```

19.配置 Nginx 错误页面优雅显示

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 .....
3 http {
4     location/ {
5         root    html/www;
6         index  index.html index.htm;
7         error_page400 401 402 403 404 405 408 410 412 413 414 415 500 501 502 503 506 = http://www.xxxx.com/er
8         # 将这些状态码的页面链接到 http://www.xxxx.com/error.html , 也可以单独指定某个状态码的页面, 如 error_page 404 /4
9     }
10 }
```

20.优化 Nginx 文件权限

为了保证网站不受木马入侵, 所有文件的用户和组都应该为 root , 所有目录的权限是 755 , 所有文件的权限是 644

```
1 [root@localhost ~]# chown -R root.root /usr/local/nginx/.... # 根据实际来调整
2 [root@localhost ~]# chmod 755 /usr/local/nginx/....
3 [root@localhost ~]# chmod 644 /usr/local/nginx/....
```

21.Nginx 防爬虫优化

我们可以根据客户端的 user-agents 首部字段来阻止指定的爬虫爬取我们的网站\

```
1 if ($http_user_agent ~* "qihoobot|Baiduspider|Googlebot|Googlebot-Mobile|Googlebot-Image|Mediapartners-Google|Adsbo
2     return 403;
```

```
3 | }
```

1. 限制单个IP的并发连接数

```
1 [root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
2 ....
3 http {
4     include      mime.types;
5     default_type  application/octet-stream;
6     sendfile      on;
7     keepalive_timeout 65;
8     limit_conn_zone $binary_remote_addr zone=addr:10m; # 用于设置共享内存区域, addr 是共享内存区域的名称, 10m 表示共
9     server {
10         listen      80;
11         server_name www.abc.com;
12         location / {
13             root     html/www;
14             index     index.html index.htm;
15             limit_conn addr1; # 限制单个IP的并发连接数为1
16         }
17     }
18 }
```

2. 限制虚拟主机总连接数

```
1 ....
2 http {
3     include      mime.types;
4     default_type  application/octet-stream;
5     sendfile      on;
6     keepalive_timeout 65;
7     limit_conn_zone $server_name zone=perserver:10m;
8     server {
```


```
9      listen      80;
10     server_name  www.abc.com;
11     location/ {
12         root      html/www;
13         index      index.html index.htm;
14         limit_conn perserver2;      # 设置虚拟主机连接数为2
15     }
16 }
17 }
```

23. 集群代理优化

```
1  upstream bbs_com_pool{ #定义服务器池
2      ip_hash;      #会话保持(当服务器集群中没有会话池时，且代理的是动态数据就必须写ip_hash,反之什么也不用写)
3      #fair      #智能分配(第三方，需要下载upstream_fair模块)根据后端服务器的响应时间来调度
4      #url_hash   #更具URL的结果来分配请求(每个url定向到同一个服务器,提高后端缓存服务器的效率，本身不支持，需要安装nginx_hash
5
6
7
8      #当算法为ip_hash不能有 weight backup
9      server192.168.10.1:80;      #默认weight为1
10     server192.168.10.3:80 weight=5;      #weight表示权重
11     server192.168.10.4:80 down;      #down:不参与本次轮询
12     server192.168.10.5:80 down backup;#backup:当任何一台主机出现故障，将进行切换替换
13     server192.168.10.6:80 max_fails=3 fail_timeout=20s;#max_fails最大失败请求次数(默认为1)，fail_timeout失败超时时间
14     server192.168.10.7:8080;
15
16 }
```

```
1  stream{
2      upstream cluster {
3          # hash $remote_addr consistent; //保持 session 不变,四层开启
4          server192.168.1.2:80 max_fails=3 fail_timeout=30s;
5          server192.168.1.3:80 max_fails=3 fail_timeout=30s;
```

```
6     }
7     server {
8         listen80;
9         proxy_pass cluster;
10        proxy_connect_timeout1s;
11        proxy_timeout3s;
12    }
13    location {
14        proxy_next_upstream http_500 http_502 http_503 error timeout invalid_header; 当发生其中任何一种错误，将转交给
15        proxy_redirect off;
16        proxy_set_header Host $host; #设置后端服务器的真实地址
17        proxy_set_header X-Real-IP $remote_addr;
18        proxy_set_header X-Forwarded_For &proxy_add_x_forwarded_for;
19        client_body_buffer_size128k; #缓冲区大小，本地保存大小
20        proxy_connect_timeout90; #发起握手等待的响应时间
21        proxy_read_timeout90; #建立连接后等待后端服务器响应时间(其实是后端等候处理的时间)
22        proxy_send_timeout90; #给定时间内后端服务器必须响应，否则断开
23        proxy_buffer_size4k; #proxy缓冲区大小
24        proxy_buffers4 32k; #缓冲区个数和大小
25        proxy_busy_buffers_size64k; #系统繁忙时buffer的临时大小，官方要求proxy_buffer_size*2
26        proxy_temp_file_write_size64k;#proxy临时文件的大小
27    }
28 }
```



```
1 vim/etc/sysctl.conf
2
3 net.ipv4.tcp_syncookies= 1
4 fs.file-max = 999999
5 net.ipv4.tcp_max_tw_buckets= 6000
6 net.ipv4.tcp_tw_reuse= 1
7 net.ipv4.tcp_tw_recycle= 1
8 net.core.somaxconn=262114
9 net.core.netdev_max_backlog=262114
```

```
10 net.ipv4.tcp_max_syn_backlog= 262114
11 net.ipv4.tcp_max_orphans=262114
12 net.ipv4.tcp_synack_retries=1
13 net.ipv4.tcp_syn_retries=1
14 net.ipv4.tcp_keepalive_time= 600
15 net.ipv4.tcp_fin_timeout= 30
16 net.ipv4.ip_local_port_range= 1024 65000
17 net.ipv4.tcp_rmem= 10240 87380 12582912
18 net.ipv4.tcp_wmem= 10240 87380 12582912
19 net.core.netdev_max_backlog= 8096
20 net.core.rmem_default= 6291456
21 net.core.wmem_default= 6291456
22 net.core.rmem_max= 12582912
23 net.core.wmem_max= 12582912
```

参数详解

[+ View Code](#)

注意：滑动窗口的大小与套接字缓存区会在一定程度上影响并发连接的数目。每个TCP连接都会为维护TCP滑动窗口而消耗内存，这个窗口会根据服务器的处理速度收缩或扩张。参数net.core.wmem_max = 12582912的设置，需要平衡物理内存的总大小、Nginx并发处理的最大连接数量而确定。当然，如果仅仅为了提供并发量使服务器不出现Out Of Memory问题而去降低滑动窗口大小，那么并不合适，因为滑动窗过小会影响大数据量的传输速度。

net.core.rmem_default = 6291456、net.core.wmem_default = 6291456、net.core.rmem_max = 12582912和net.core.wmem_max = 12582912这4个参数的设置需要根据我们的业务特性以及实际的硬件成本来综合考虑。Nginx并发处理的最大连接量：由nginx.conf中的work_processes和work_connections参数决定。

分类: [nginx](#)、[tomcat](#)、[apache](#)

[好文要顶](#)[关注我](#)[收藏该文](#)

小雨浙浙o0

粉丝 - 40 关注 - 18

[+加关注](#)

0

0

« 上一篇: kubernetes的简介及使用教程 (转)

» 下一篇: Tomcat调优

posted @ 2020-01-05 21:25 小雨浙浙o0 阅读(451) 评论(0) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论, 立即 [登录](#) 或者 [逛逛](#) 博客园首页

【推荐】园子的商业化努力-AI人才服务: 招募AI导师, 一起探索AI领域的机会

【推荐】中国电信天翼云: 算力普惠, 释放红利, 核心产品让利90%低至1折

【推荐】中国云计算领导者: 阿里云轻量应用服务器2核2G低至108元/年

【推荐】第五届金蝶云苍穹低代码开发大赛正式启动, 百万奖金等你拿!

编辑推荐:

- 记一次 .NET 某医院门诊软件 卡死分析
- 我试图通过这篇文章告诉你, 这行源码有多牛逼
- [MAUI 程序设计] 界面多态与实现
- 记一次 某智能制造MES系统CPU 爆高分析
- 一个操作把 MySQL 主从复制整崩了

即构专区：

- 多路混流实操流程
- 即构实时音视频多中心调度设计
- ZEGO即构自建MSDN有序网络，为实时音视频传输极致顺畅！
- 影响音视频延迟的关键因素（三）： 传输、渲染
- 即构低延迟直播产品L3，打造更优质的实时互动体验

Copyright © 2023 小雨浙浙o0

Powered by .NET 7.0 on Kubernetes