nginx优化 突破十万并发

文章转载于：http://9388751.blog.51cto.com/9378751/1676821

nginx优化 突破十万并发

一、一般来说nginx 配置文件中对优化比较有作用的为以下几项：

1. worker_processes 8;

nginx 进程数，建议按照cpu 数目来指定，一般为它的倍数 (如,2个四核的cpu计为8)。

2. worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;

为每个进程分配cpu，上例中将8 个进程分配到8 个cpu，当然可以写多个，或者将一
个进程分配到多个cpu。

3. worker_rlimit_nofile 65535;

这个指令是指当一个nginx 进程打开的最多文件描述符数目，理论值应该是最多打开文
件数（ulimit -n）与nginx 进程数相除，但是nginx 分配请求并不是那么均匀，所以最好与ulimit -n 的值保持一致。

现在在linux 2.6内核下开启文件打开数为65535，worker_rlimit_nofile就相应应该填写65535。

这是因为nginx调度时分配请求到进程并不是那么的均衡，所以假如填写10240，总并发量达到3-4万时就有进程可能超过10240了，这时会返回502错误。

查看linux系统文件描述符的方法：

[root@web001 ~]# sysctl -a | grep fs.file

fs.file-max = 789972

fs.file-nr = 510 0 789972

4. use epoll;

使用epoll 的I/O 模型

(

补充说明:

与apache相类，nginx针对不同的操作系统，有不同的事件模型

A）标准事件模型
Select、poll属于标准事件模型，如果当前系统不存在更有效的方法，nginx会选择select或poll
B）高效事件模型
Kqueue：使用于 FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X. 使用双处理器的MacOS X系统使用kqueue可能会造成内核崩溃。
Epoll: 使用于Linux内核2.6版本及以后的系统。

/dev/poll：使用于 Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ 和 Tru64 UNIX 5.1A+。

Eventport：使用于 Solaris 10. 为了防止出现内核崩溃的问题， 有必要安装安全补丁。

)

5. worker_connections 65535;

每个进程允许的最多连接数， 理论上每台nginx 服务器的最大连接数为worker_processes*worker_connections。

6. keepalive_timeout 60;

keepalive 超时时间。

7. client_header_buffer_size 4k;

客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个请求头的大小不会超过1k，不过由于一般系统分页都要大于1k，所以这里设置为分页大小。

分页大小可以用命令getconf PAGESIZE 取得。

[root@web001 ~]# getconf PAGESIZE

4096

但也有client_header_buffer_size超过4k的情况，但是client_header_buffer_size该值必须设置为“系统分页大小”的整倍数。

8. open_file_cache max=65535 inactive=60s;

这个将为打开文件指定缓存，默认是没有启用的，max 指定缓存数量，建议和打开文件数一致，inactive 是指经过多长时间文件没被请求后删除缓存。

9. open_file_cache_valid 80s;

这个是指多长时间检查一次缓存的有效信息。

10. open_file_cache_min_uses 1;

open_file_cache 指令中的inactive 参数时间内文件的最少使用次数，如果超过这个数字，文件描述符一直是在缓存中打开的，如上例，如果有一个文件在inactive 时间内一次没被使用，它将被移除。

二、关于内核参数的优化：

net.ipv4.tcp_max_tw_buckets = 6000

timewait 的数量，默认是180000。

net.ipv4.ip_local_port_range = 1024 65000

允许系统打开的端口范围。

net.ipv4.tcp_tw_recycle = 1

启用timewait 快速回收。

net.ipv4.tcp_tw_reuse = 1

开启重用。允许将TIME-WAIT sockets 重新用于新的TCP 连接。

net.ipv4.tcp_syncookies = 1

开启SYN Cookies，当出现SYN 等待队列溢出时，启用cookies 来处理。

net.core.somaxconn = 262144

web 应用中listen 函数的backlog 默认会给我们内核参数的net.core.somaxconn 限制到128，而nginx 定义的NGX_LISTEN_BACKLOG 默认为511，所以有必要调整这个值。

net.core.netdev_max_backlog = 262144

每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目。

net.ipv4.tcp_max_orphans = 262144

系统中最多有多少个TCP 套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤儿连接将即刻被复位并打印出警告信息。这个限制仅仅是为了防止简单的DoS 攻击，不能过分依靠它或者人为地减小这个值，更应该增加这个值(如果增加了内存之后)。

net.ipv4.tcp_max_syn_backlog = 262144

记录的那些尚未收到客户端确认信息的连接请求的最大值。对于有128M 内存的系统而言，缺省值是1024，小内存的系统则是128。

net.ipv4.tcp_timestamps = 0

时间戳可以避免序列号的卷绕。一个1Gbps 的链路肯定会遇到以前用过的序列号。时间戳能够让内核接受这种“异常”的数据包。这里需要将其关掉。

net.ipv4.tcp_synack_retries = 1

为了打开对端的连接，内核需要发送一个SYN 并附带一个回应前面一个SYN的ACK。也就是所谓三次握手中的第二次握手。这个设置决定了内核放弃连接之前发送SYN+ACK 包的数量。

net.ipv4.tcp_syn_retries = 1

在内核放弃建立连接之前发送SYN 包的数量。

net.ipv4.tcp_fin_timeout = 1

如 果套接字由本端要求关闭，这个参数决定了它保持在FIN-WAIT-2 状态的时间。对端可以出错并永远不关闭连接，甚至意外当机。缺省值是60 秒。2.2 内核的通常值是180 秒，3你可以按这个设置，但要记住的是，即使你的机器是一个轻载的WEB 服务器，也有因为大量的死套接字而内存溢出的风险，FIN- WAIT-2 的危险性比FIN-WAIT-1 要小，因为它最多只能吃掉1.5K 内存，但是它们的生存期长些。

net.ipv4.tcp_keepalive_time = 30

当keepalive 起用的时候，TCP 发送keepalive 消息的频度。缺省是2 小时。

三、下面贴一个完整的内核优化设置:

vi /etc/sysctl.conf CentOS5.5中可以将所有内容清空直接替换为如下内容:

net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1

net.ipv4.tcp_syncookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
net.ipv4.tcp_max_tw_buckets = 6000
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_rmem = 4096 87380 4194304
net.ipv4.tcp_wmem = 4096 16384 4194304
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.netdev_max_backlog = 262144
net.core.somaxconn = 262144
net.ipv4.tcp_max_orphans = 3276800
net.ipv4.tcp_max_syn_backlog = 262144
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 1
net.ipv4.tcp_syn_retries = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_fin_timeout = 1
net.ipv4.tcp_keepalive_time = 30
net.ipv4.ip_local_port_range = 1024 65000

使配置立即生效可使用如下命令:
/sbin/sysctl -p

四、下面是关于系统连接数的优化

linux 默认值 open files 和 max user processes 为 1024

#ulimit -n

1024

#ulimit Cu

1024

问题描述： 说明 server 只允许同时打开 1024 个文件，处理 1024 个用户进程

使用ulimit -a 可以查看当前系统的所有限制值，使用ulimit -n 可以查看当前的最大打开文件数。

新装的linux 默认只有1024 ，当作负载较大的服务器时，很容易遇到error: too many open files 。因此，需要将其改大。

解决方法：

使用 ulimit Cn 65535 可即时修改，但重启后就无效了。（注ulimit -SHn 65535 等效 ulimit -n 65535 ，-S 指soft ，-H 指hard)

有如下三种修改方式：

1. 在/etc/rc.local 中增加一行 ulimit -SHn 65535
2. 在/etc/profile 中增加一行 ulimit -SHn 65535
3. 在/etc/security/limits.conf 最后增加：

* soft nofile 65535
* hard nofile 65535
* soft nproc 65535
* hard nproc 65535

具体使用哪种，在 CentOS 中使用第1 种方式无效果，使用第3 种方式有效果，而在Debian 中使用第2 种有效果

ulimit -n

65535

ulimit -u

65535

备注：ulimit 命令本身就有分软硬设置，加-H 就是硬，加-S 就是软默认显示的是软限制

soft 限制指的是当前系统生效的设置值。 hard 限制值可以被普通用户降低。但是不能增加。 soft 限制不能设置的比 hard 限制更高。 只有 root 用户才能够增加 hard 限制值。

五、下面是一个简单的nginx 配置文件：

```
user www www;
worker_processes 8;
worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000
00100000
01000000;
error_log /www/log/nginx_error.log crit;
pid /usr/local/nginx/nginx.pid;
worker_rlimit_nofile 204800;
events
{
use epoll;
worker_connections 204800;
}
http
{
include mime.types;
default_type application/octet-stream;
charset utf-8;
server_names_hash_bucket_size 128;
client_header_buffer_size 2k;
large_client_header_buffers 4 4k;
client_max_body_size 8m;
sendfile on;
tcp_nopush on;
keepalive_timeout 60;
```

```
fastcgi_cache_path /usr/local/nginx/fastcgi_cache levels=1:2
keys_zone=TEST:10m
inactive=5m;
fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;
fastcgi_buffer_size 4k;
fastcgi_buffers 8 4k;
fastcgi_busy_buffers_size 8k;
fastcgi_temp_file_write_size 8k;
fastcgi_cache TEST;
fastcgi_cache_valid 200 302 1h;
fastcgi_cache_valid 301 1d;
fastcgi_cache_valid any 1m;
fastcgi_cache_min_uses 1;
fastcgi_cache_use_stale error timeout invalid_header http_500;
open_file_cache max=204800 inactive=20s;
open_file_cache_min_uses 1;
open_file_cache_valid 30s;
tcp_nodelay on;
gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.0;
gzip_comp_level 2;
gzip_types text/plain application/x-javascript text/css
application/xml;
gzip_vary on;
server
{
listen 8080;
server_name backup.aiju.com;
index index.php index.htm;
root /www/html/;
```

```
location /status
{
stub_status on;
}
location ~ .*/.(php|php5)?$
{
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
include fcgi.conf;
}
location ~ .*/.(gif|jpg|jpeg|png|bmp|swf|js|css)$
{
expires 30d;
}
log_format access ‘$remote_addr — $remote_user [$time_local]
 “$request” ‘
 ‘$status $body_bytes_sent “$http_referer” ‘
 ‘” $http_user_agent” $http_x_forwarded_for’;
access_log /www/log/access.log access;
}
}
```

六、关于FastCGI 的几个指令：

```
fastcgi_cache_path /usr/local/nginx/fastcgi_cache levels=1:2
keys_zone=TEST:10minactive=5m;
```

这个指令为FastCGI 缓存指定一个路径，目录结构等级，关键字区域存储时间和非活动删除时间。

```
fastcgi_connect_timeout 300;
```

指定连接到后端FastCGI 的超时时间。

```
fastcgi_send_timeout 300;
```

向FastCGI 传送请求的超时时间，这个值是指已经完成两次握手后向

FastCGI 传送请求的超时时间。

fastcgi_read_timeout 300;

接收FastCGI 应答的超时时间，这个值是指已经完成两次握手后接收FastCGI 应答的超时时间。

fastcgi_buffer_size 4k;

指定读取FastCGI 应答第一部分需要用多大的缓冲区，一般第一部分应答不会超过1k，由于页面大小为4k，所以这里设置为4k。

fastcgi_buffers 8 4k;

指定本地需要用多少和多大的缓冲区来缓冲FastCGI 的应答。

fastcgi_busy_buffers_size 8k;

这个指令我也不知道是做什么用，只知道默认值是fastcgi_buffers 的两倍。

fastcgi_temp_file_write_size 8k;

在写入fastcgi_temp_path 时将用多大的数据块，默认值是fastcgi_buffers 的两倍。

fastcgi_cache TEST

开启FastCGI 缓存并且为其制定一个名称。个人感觉开启缓存非常有用，可以有效降低CPU 负载，并且防止502 错误。

fastcgi_cache_valid 200 302 1h;
fastcgi_cache_valid 301 1d;
fastcgi_cache_valid any 1m;

为指定的应答代码指定缓存时间，如上例中将200，302 应答缓存一小时，301 应答缓存1 天，其他为1 分钟。

fastcgi_cache_min_uses 1;

缓存在fastcgi_cache_path 指令inactive 参数值时间内的最少使用次数，如上例，如果在5 分钟内某文件1 次也没有被使用，那么这个文件将被移除。

fastcgi_cache_use_stale error timeout invalid_header http_500;

不知道这个参数的作用，猜想应该是让nginx 知道哪些类型的缓存是没用的。以上为nginx 中FastCGI 相关参数，另外，FastCGI 自身也有一些配置需要进行优化，如果你使用php-fpm 来管理FastCGI，可以修改配置文件中的以下值：

<value name=”max_children”>60</value>

同时处理的并发请求数，即它将开启最多60 个子线程来处理并发连接。

<value name=”rlimit_files”>102400</value>

最多打开文件数。

<value name=”max_requests”>204800</value>

每个进程在重置之前能够执行的最多请求数。