一、centos7编译安装nginx 1.16.1稳定版

1.1 下载地址

已上传140SVN

```
http://39.100.254.140:12011/loit-Infrastructure-doc/loit-initproject-doc/blob/master/3%E3%80%81other/tools/nginx-1.16.1.tar.gz

http://39.100.254.140:12011/loit-Infrastructure-doc/loit-initproject-doc/blob/master/3%E3%80%81other/tools/echo-nginx-module-0.61.tar.gz
```

nginx-1.16.1.tar.gz 上传到目录：/usr/local/src echo-nginx-module-0.61.tar.gz 上传到root/echo-nginx-module-0.61.tar.gz 并解压

```
tar -zxvf echo-nginx-module-0.61.tar.gz
```

1.2 安装nginx

安装`nginx`编译所需的lib库

```
yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel
yum -y install pcre pcre-devel
```

查看`pcre`(正则库)版本

```
pcre-config --version

8.32
```

进入编译目录

```
cd /usr/local/src
```

解压nginx压缩包

```
tar -zxvf nginx-1.16.1.tar.gz
```

进入解压目录

```
cd  nginx-1.16.1
```

运行配置脚本(--prefix参数指定nginx安装的目录,默认安装在/usr/local/nginx)

```
./configure --prefix=/usr/local/nginx --add-module=/root/echo-nginx-module-0.61 --with-
http_stub_status_module
```

编译安装nginx

```
make && make install
```

将`nginx`执行命令软链接到`/usr/bin`

```
ln -s /usr/local/nginx/sbin/nginx /usr/bin
```

启动nginx

```
nginx
```

设置开机自启动

```
echo "/usr/local/nginx/sbin/nginx" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local
```

1.3 测试echo模块

```
location /hello {
    default_type 'text/plain';
    return 200 'hello!';
}

location /hello_echo {
    default_type 'text/plain';
    echo "hello, echo!";
}
```

```
curl http://127.0.0.1/hello
```

1.4 stub_status模块用法

提供了查看 Nginx 运行的基本状态信息，我们只想让部分 IP 的人可以查看，此时可以配置一个访问控制：

```
vi /usr/local/nginx/conf/nginx.conf
```

```
location /nginx-status {
    stub_status;
    access_log off;
    #allow 192.168.179.0/24;
    #deny all;
}
```

```
curl 127.0.0.1/nginx-status
```

1.4 nginx相关命令

执行`nginx -h`查看相关命令

```
[root@localhost ~]# nginx -h
nginx version: nginx/1.16.1
Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit
  -t            : test configuration and exit
  -T            : test configuration, dump it and exit
  -q            : suppress non-error messages during configuration testing
  -s signal     : send signal to a master process: stop, quit, reopen, reload
  -p prefix     : set prefix path (default: /usr/local/nginx/)
  -c filename   : set configuration file (default: conf/nginx.conf)
  -g directives : set global directives out of configuration file
复制代码
```

查看nginx安装目录

```
whereis nginx
```

停止重启

```
启动
[root sbin]# ./nginx
停止
[root sbin]# ./nginx -s stop
重启
[root sbin]# ./nginx -s reload
```

开启端口80

```
firewall-cmd --zone=public --add-port=80/tcp --permanent
```

命令含义:

–zone #作用域

–add-port=80/tcp #添加端口，格式为：端口/通讯协议

–permanent #永久生效，没有此参数重启后失效

重启防火墙

```
firewall-cmd --reload #重启firewall
systemctl stop firewalld.service #停止firewall
systemctl disable firewalld.service #禁止firewall开机启动
firewall-cmd --state #查看默认防火墙状态（关闭后显示notrunning，开启后显示running）
```

1.5 测试工具

1、ab 测试工具安装

```
yum -y install httpd-tools
```

测试2000连接数，50000次请求

```
ab -c 2000 -n 50000 http://172.16.203.78/hello
```

2、wrk测试工具

```
wrk -t50 -c300 -d30s -T30 http://172.16.203.78/hello
```

二、性能问题

2.1 未优化前

测试环境

```
虚拟机
cpu 核数: 8核
内存: 4G
```

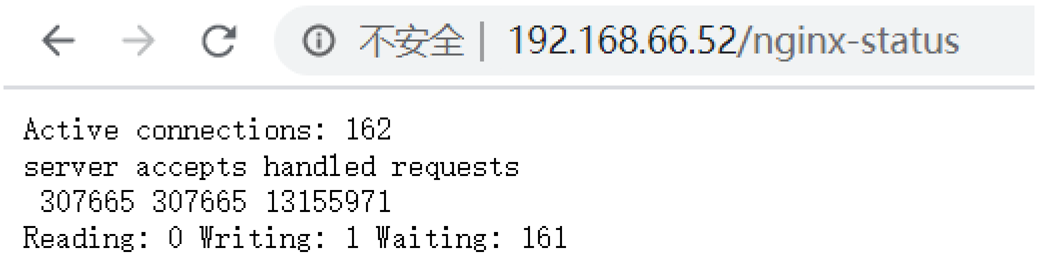
查看当前cpu的状态:

```
[root ~]# lscpu |grep "CPU(s)"
```

🚚 1000 并发

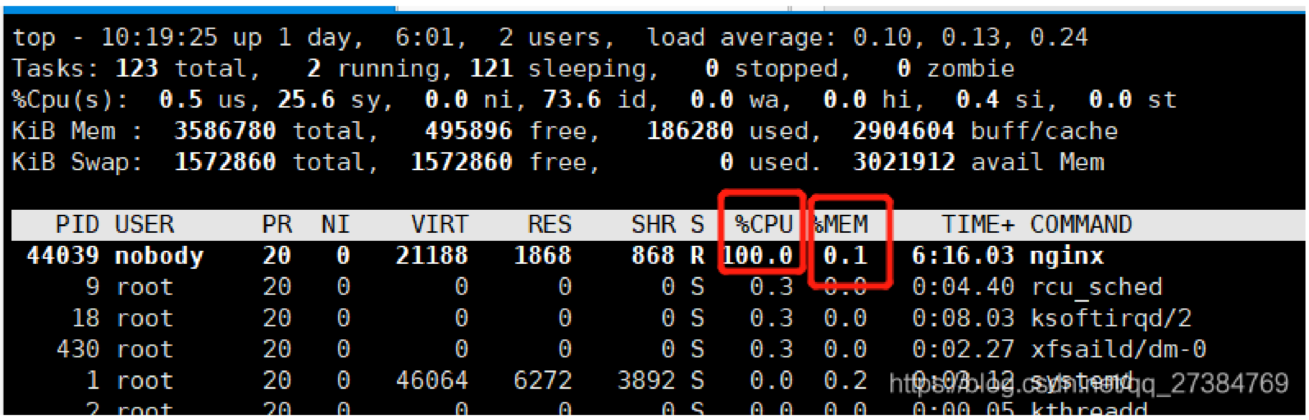
```
wrk -t50 -c1000 -d30s -T30 http://172.16.203.78/hello

Running 30s test @ http://192.168.66.52/hello
  50 threads and 1000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    38.67ms  229.80ms   6.47s    97.16%
    Req/Sec     1.90k     1.46k   17.12k    76.88%
  2609441 requests in 30.08s, 380.63MB read
  Socket errors: connect 29, read 0, write 0, timeout 0
Requests/sec:  86757.38
Transfer/sec:     12.65MB
```

- QPS 为 **86757.38**
- 平均延迟为 38.67ms
- 其中出现 **Socket errors**: connect 29



```
Active connections: 162
server accepts handled requests
 307665 307665 13155971
Reading: 0 Writing: 1 Waiting: 161
```

- Active connections: 在200左右比较低
- Waiting数量比较多



```
top - 10:19:25 up 1 day,  6:01,  2 users,  load average: 0.10, 0.13, 0.24
Tasks: 123 total,   2 running, 121 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.5 us, 25.6 sy,  0.0 ni, 73.6 id,  0.0 wa,  0.0 hi,  0.4 si,  0.0 st
KiB Mem :  3586780 total,   495896 free,   186280 used,  2904604 buff/cache
KiB Swap:  1572860 total,  1572860 free,        0 used.  3021912 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
44039 nobody    20   0   21188   1868    868 R 100.0  0.1   6:16.03 nginx
    9 root      20   0       0      0      0 S   0.3  0.0   0:04.40 rcu_sched
   18 root      20   0       0      0      0 S   0.3  0.0   0:08.03 ksoftirqd/2
  430 root      20   0       0      0      0 S   0.3  0.0   0:02.27 xfsaild/dm-0
    1 root      20   0   46064   6272   3892 S   0.0  0.2   0:03.12 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.05 kthreadd
```

- CPU使用率 100%
- 内存0.1%

🚚 1000 并发

```
wrk -t50 -c2000 -d30s -T30 http://192.168.66.52/hello
unable to create thread 28: Too many open files
```

查看nginx错误日志

```
# tail /usr/local/nginx/logs/error.log
```

```
2020/08/07 08:33:30 [error] 44039#0: *59630 open() "/usr/local/nginx/html/favicon.ico"
failed (2: No such file or directory), client: 192.168.66.240, server: localhost, request:
"GET /favicon.ico HTTP/1.1", host: "192.168.66.52"
2020/08/07 08:48:19 [crit] 44039#0: accept4() failed (24: Too many open files)
```

- 出现错误 Too many open files

2.2 问题总结

- 并发1000出现socket异常、nginx Active 数量少、Waiting数量多。
- 并发2000出现 Too many open files 异常。

三、优化思路

> 1、系统和nginx是否可以建立多个socket连接
>
> 2、系统和nginx是否允许一次性打开多个文件

建立socket连接，从操作系统和nginx两个层面分析

(1) 从nginx

> 1、http连接快速关闭即配置nginx的 keep_alivetime:0。因为在HTTP 1.0中协议是 请求-》连接-》断开，即每次请求之后都需要再次握手，但是随着web应用的丰富出现很多css文件和其他资源文件，这就使得要求是否一次请求可以请求多个文件，这就是HTTP 1.1。
>
> 2、子进程允许打开的连接即配置nginx的（worker_connections）

(2) 从系统层面：

> (1)修改最大连接数 somaxconn(具体路径在 /proc/sys/net/core/somaxconn) (2)加快tcp连接的回收，即修改（/proc/sys/net/ipv4/tcp_tw_recycle） (3)空闲的tcp是否允许回收利用，即修改（/proc/sys/net/ipv4/tcp_tw_reuse） (4)是否对洪水抵御做相应的cookie操作，修改（/proc/sys/net/ipv4/tcp_syncookies）

打开文件方面

1.nginx: 子进程允许打开的文件数量：配置添加：worker_rlimit_nofile 2.系统：设置ulimit -n 设置一个较大的值

一、 最大打开文件数的限制

```
vi /etc/security/limits.conf

最后添加

# End of file
root soft nofile 65535
root hard nofile 65535
* soft nofile 65535
* hard nofile 65535
```

二、用户进程限制

```
vi /etc/security/limits.d/20-nproc.conf

  #加大普通用户限制   也可以改为unlimited
  *          soft    nproc     40960
  root       soft    nproc     unlimited
```

四、优化

内核参数:

vi /etc/sysctl.conf

```
net.ipv4.conf.default.rp_filter = 1

net.ipv4.ip_forward = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
net.ipv4.ip_local_port_range = 1024 65535
net.ipv4.tcp_max_syn_backlog = 65535

net.ipv4.tcp_max_tw_buckets = 262144

net.core.somaxconn = 65535
net.core.netdev_max_backlog = 200000
net.core.rmem_default = 67108864
net.core.wmem_default = 67108864
net.core.rmem_max = 67108864
net.core.wmem_max = 67108864
net.ipv4.tcp_rmem = 4096 87380 6291456
net.ipv4.tcp_wmem = 4096 65536 4194304
net.ipv4.tcp_mem = 3097431 4129911 6194862
net.ipv4.tcp_timestamps = 0

net.ipv4.tcp_syncookies = 1

net.ipv4.tcp_synack_retries = 1
net.ipv4.tcp_syn_retries = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 0
net.ipv4.ip_forward = 1
net.ipv4.tcp_fin_timeout = 15
net.ipv4.tcp_keepalive_time = 120
vm.overcommit_memory = 1
fs.file-max = 1048576
```

运行 sysctl -p后配置生效

```
sysctl -p
```

部分参数说明

```
net.ipv4.tcp_syncookies = 1  表示开启SYN Cookies。当出现SYN等待队列溢出时，启用cookies来处理，可防范少量SYN攻击，默认为0，表示关闭；
net.ipv4.tcp_tw_reuse = 1  表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接，默认为0，表示关闭；
net.ipv4.tcp_fin_timeout = 720  表示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。
```

Nginx 参考配置文件如下：

```
user root;
worker_processes  10;
#daemon off;
#master_process  off;

worker_cpu_affinity
#000000000001
#000000000010
000000000100
000000001000
000000010000
000000100000
000001000000
000010000000
```

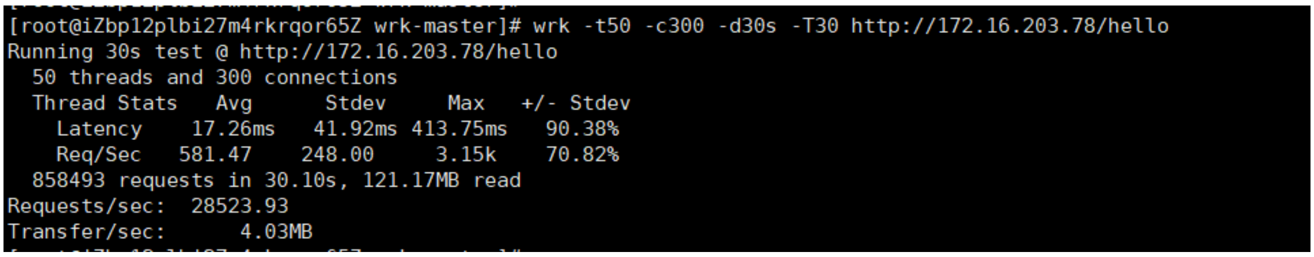
```
000100000000
001000000000
010000000000
100000000000
;

#error_log   logs/error.log debug;
error_log   logs/error.log;

worker_rlimit_core 200m;
working_directory /tmp;

pid             logs/nginx.pid;
events {
                worker_connections 204800;
        use epoll;
        accept_mutex off;
        multi_accept on;
}

http {
    sendfile        on;
    tcp_nodelay        on;
    tcp_nopush  on;

        access_log off;

        server_tokens off;
        reset_timedout_connection on;

        keepalive_timeout 120;
        keepalive_requests 100000;

        client_max_body_size 20m;
        client_body_buffer_size 1024k;
        client_body_temp_path /tmp;

        upstream redis_cluster {
                testupstream_node $node_ip;
                server 0.0.0.0;
                keepalive 1024;
        }

        server {
        listen   80;
        server_name  localhost backlog=204800;

                set $backserver "redis_cluster";
                set $node_ip "";

                location ~* "^/hdp/kvstore/" {
                        testupstream_pass $backserver;
                        testupstream_next_upstream error timeout invalid_response;
        }

                location /hello {
            default_type 'text/plain';
            return 200 'hello!';
         }

        location /hello_echo {
            default_type 'text/plain';
            echo "hello, echo!";
         }


        location /nginx-status {
             stub_status;
             access_log off;
             #allow 192.168.179.0/24;
             #deny all;
        }

    }
}
```

五、优化后测试

服务器配置

```
cpu 8核
内存 32G
```

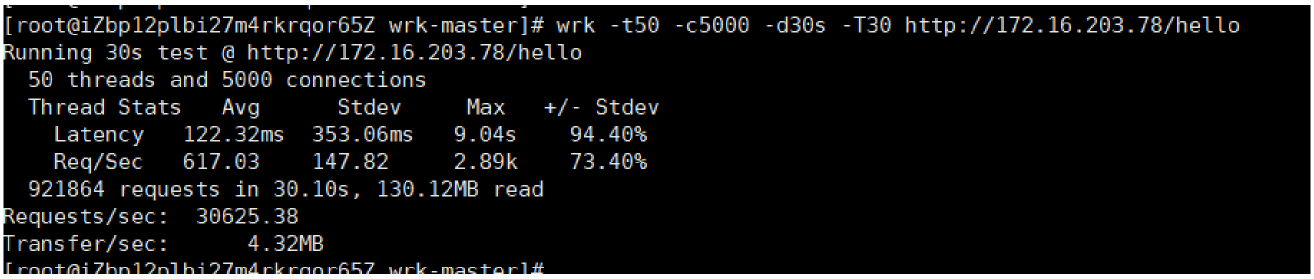
🚚 300 并发



```
[root@iZbp12plbi27m4rkrqor65Z wrk-master]# wrk -t50 -c300 -d30s -T30 http://172.16.203.78/hello
Running 30s test @ http://172.16.203.78/hello
  50 threads and 300 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    17.26ms   41.92ms 413.75ms   90.38%
    Req/Sec   581.47    248.00     3.15k    70.82%
  858493 requests in 30.10s, 121.17MB read
Requests/sec:  28523.93
Transfer/sec:      4.03MB
```

- QPS 为 **28523.93**
- 平均延迟为 17.26ms

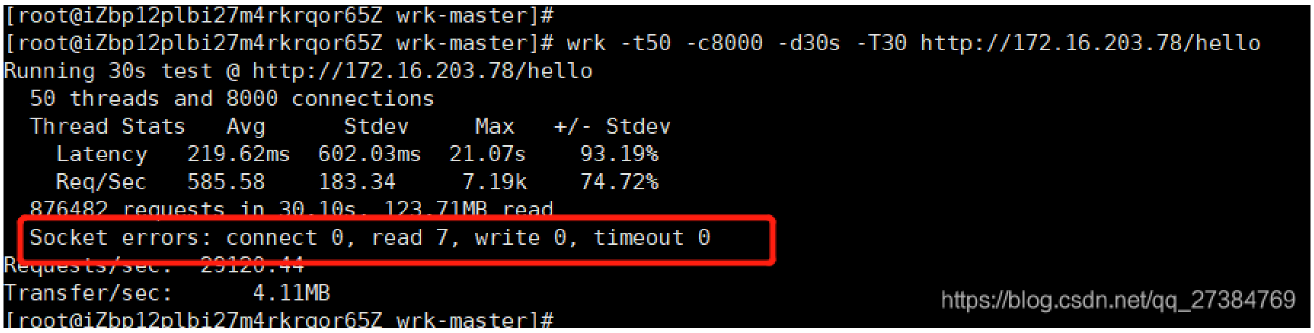
5000 并发



```
[root@iZbp12plbi27m4rkrqor65Z wrk-master]# wrk -t50 -c5000 -d30s -T30 http://172.16.203.78/hello
Running 30s test @ http://172.16.203.78/hello
  50 threads and 5000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   122.32ms  353.06ms   9.04s    94.40%
    Req/Sec   617.03    147.82     2.89k    73.40%
  921864 requests in 30.10s, 130.12MB read
Requests/sec:  30625.38
Transfer/sec:      4.32MB
[root@iZbp12plbi27m4rkrqor65Z wrk-master]#
```

- QPS 为 **30625.28**
- 平均延迟为 122.32ms

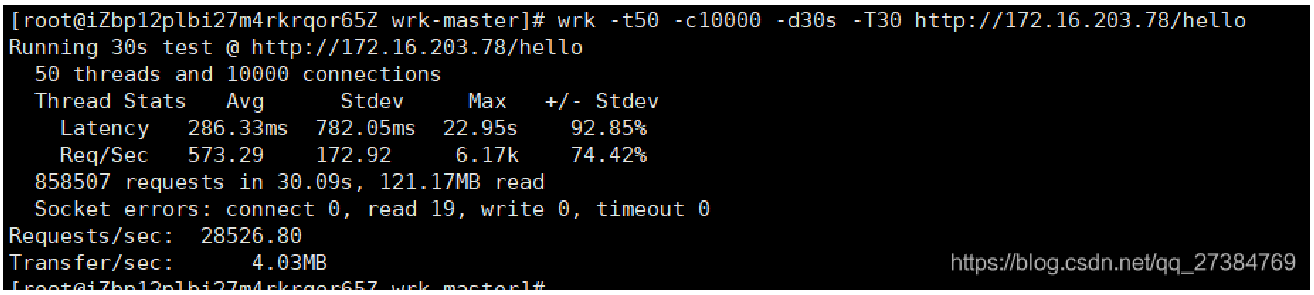
8000 并发



```
[root@iZbp12plbi27m4rkrqor65Z wrk-master]#
[root@iZbp12plbi27m4rkrqor65Z wrk-master]# wrk -t50 -c8000 -d30s -T30 http://172.16.203.78/hello
Running 30s test @ http://172.16.203.78/hello
  50 threads and 8000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   219.62ms  602.03ms  21.07s    93.19%
    Req/Sec   585.58    183.34     7.19k    74.72%
  876482 requests in 30.10s, 123.71MB read
  Socket errors: connect 0, read 7, write 0, timeout 0
Requests/sec:  29120.44
Transfer/sec:      4.11MB
[root@iZbp12plbi27m4rkrqor65Z wrk-master]#
```

- QPS 为 **29120.44**
- 平均延迟为 219.62ms
- socket errors: **read 7**

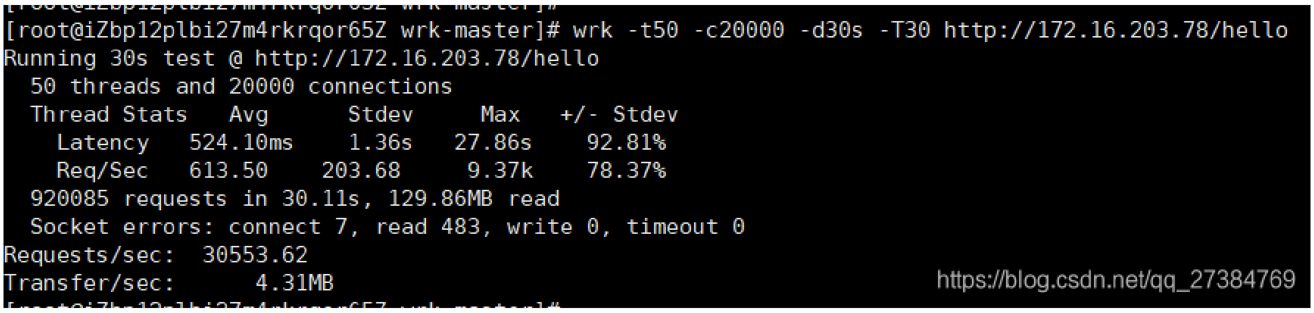
10000 并发



```
[root@iZbp12plbi27m4rkrqor65Z wrk-master]# wrk -t50 -c10000 -d30s -T30 http://172.16.203.78/hello
Running 30s test @ http://172.16.203.78/hello
  50 threads and 10000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   286.33ms  782.05ms  22.95s    92.85%
    Req/Sec   573.29    172.92     6.17k    74.42%
  858507 requests in 30.09s, 121.17MB read
  Socket errors: connect 0, read 19, write 0, timeout 0
Requests/sec:  28526.80
Transfer/sec:      4.03MB
[root@iZbp12plbi27m4rkrqor65Z wrk-master]#
```

- QPS 为 **28526**
- 平均延迟为 286.33ms
- socket errors: **read 19**

20000 并发

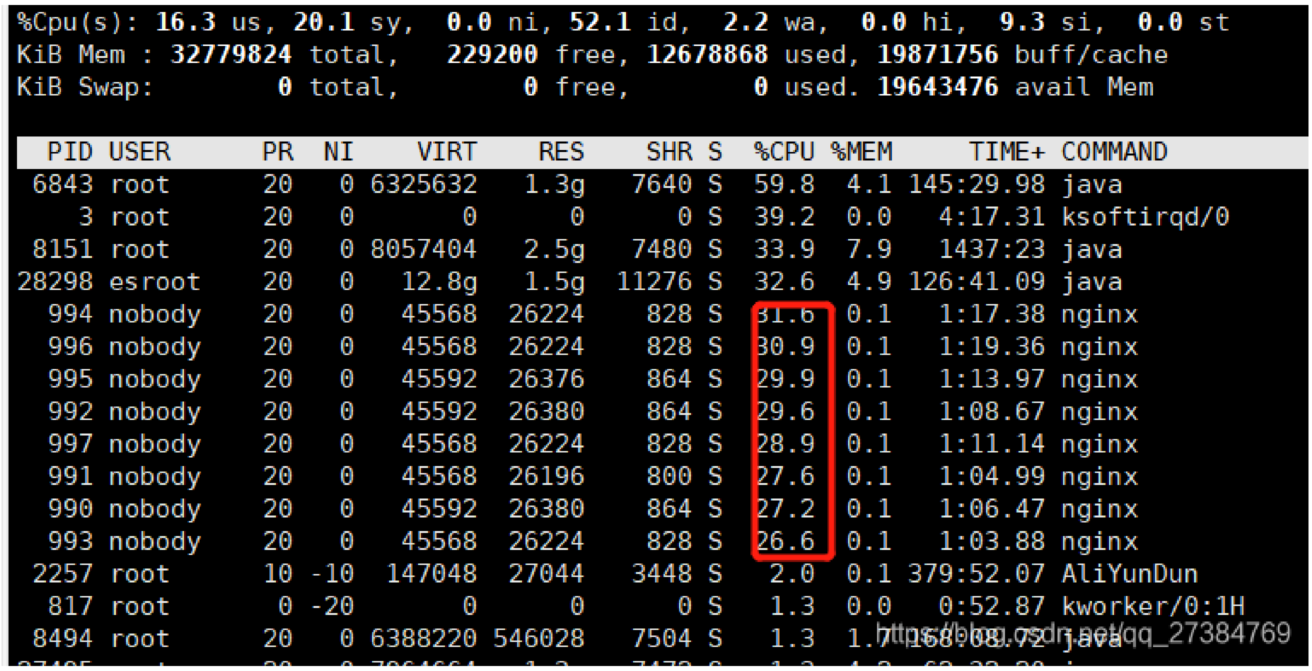


```
[root@iZbp12plbi27m4rkrqor65Z wrk-master]#
[root@iZbp12plbi27m4rkrqor65Z wrk-master]# wrk -t50 -c20000 -d30s -T30 http://172.16.203.78/hello
Running 30s test @ http://172.16.203.78/hello
  50 threads and 20000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   524.10ms    1.36s   27.86s    92.81%
    Req/Sec   613.50    203.68     9.37k    78.37%
  920085 requests in 30.11s, 129.86MB read
  Socket errors: connect 7, read 483, write 0, timeout 0
Requests/sec:  30553.62
Transfer/sec:      4.31MB
[root@iZbp12plbi27m4rkrqor65Z wrk-master]#
```

- QPS 为 **30553.62**

- 平均延迟为 286.33ms
- socket errors: **connect 7 read 483**

CPU使用情况



```
%Cpu(s): 16.3 us, 20.1 sy,  0.0 ni, 52.1 id,  2.2 wa,  0.0 hi,  9.3 si,  0.0 st
KiB Mem : 32779824 total,   229200 free, 12678868 used, 19871756 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 19643476 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 6843 root      20   0 6325632   1.3g   7640 S  59.8  4.1 145:29.98 java
    3 root      20   0       0      0      0 S  39.2  0.0   4:17.31 ksoftirqd/0
 8151 root      20   0 8057404   2.5g   7480 S  33.9  7.9   1437:23 java
28298 esroot    20   0   12.8g   1.5g  11276 S  32.6  4.9 126:41.09 java
  994 nobody    20   0   45568  26224    828 S  31.6  0.1   1:17.38 nginx
  996 nobody    20   0   45568  26224    828 S  30.9  0.1   1:19.36 nginx
  995 nobody    20   0   45592  26376    864 S  29.9  0.1   1:13.97 nginx
  992 nobody    20   0   45592  26380    864 S  29.6  0.1   1:08.67 nginx
  997 nobody    20   0   45568  26224    828 S  28.9  0.1   1:11.14 nginx
  991 nobody    20   0   45568  26196    800 S  27.6  0.1   1:04.99 nginx
  990 nobody    20   0   45592  26380    864 S  27.2  0.1   1:06.47 nginx
  993 nobody    20   0   45568  26224    828 S  26.6  0.1   1:03.88 nginx
 2257 root      10 -10  147048  27044   3448 S   2.0  0.1 379:52.07 AliYunDun
  817 root       0 -20       0      0      0 S   1.3  0.0   0:52.87 kworker/0:1H
 8494 root      20   0 6388220 546028   7504 S   1.3  1.7 168:08.72 java
```

- 测试过程平均cpu使用率40%

六、nginx性能测试结论

nginx优化的方法三种，第一种优化linux内核参数，使内核变的更为强大，第二种是优化nginx配置文件，使nginx变的更为强大, 第三种是扩展服务器的cpu和内存，使服务器变的更为强大。

单机测试:

- 单机8核cpu的平均在30000QPS, 1万并发连接数平均消耗40%cpu。
- nginx并发数与**cpu核数**有关，cpu核数到达**88核**可以实现百万QPS数量。
- 并发连接数达到8000 ~ 10000 开始有很少量的error，并发连接数达到20000 error 数量开始上升。

参考内存配置要求:

在操作系统层面每个TCP连接会占用3k-10k的内存，以20万来计算，需要2G内存。nginx程序本身还要消耗内存，特别是nginx反向代理POST请求比较多的情况，20万连接情况下推荐16G内存配置。