压力测试参数：
方案 1：
　　线程：100　　测试时长：1min
方案 2：
　　线程：300　　测试时长：2min
方案 3：
　　线程：300（分布式 3 台）　测试时长：2min

系统优化

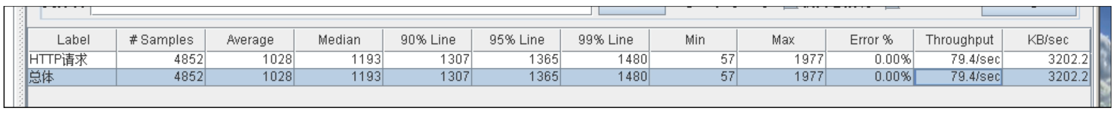
做一些基本的安全配置及优化，形成脚本自动化配置

```
#cat linux_centos6_base_web.sh
#!/bin/bash
#*****************************************************************#
# ScriptName: linux_centos6_base.sh
# Author: liujmsunits@hotmail.com
# Create Date: 2015-07-13 11:38
# Modify Author: liujmsunits@hotmail.com
# Modify Date: 2015-07-13 15:38
# Function:web 系统基础优化
#*****************************************************************#
set -x
function deterpm()
{
	rpm -qa |grep $1
}

function Decide()
{
	if [ $1 = 0 ];then
		:
	else
		$2
	fi
}

function ntpconfig()
```

```
{
        /etc/init.d/ntpd stop
        /usr/sbin/ntpdate ntp.api.bz > /dev/null 2>&1
        /etc/init.d/ntpd start
        echo "*/5 * * * * /usr/sbin/ntpdate ntp.api.bz > /dev/null 2>&1" >> /var/spool/cron/root
}
#source
wget  -c  http://mirrors.163.com/.help/CentOS6-Base-163.repo  -O  /etc/yum.repos.d/CentOS6-Base-163.repo
yum clean metadata
nohup yum makecache &

#ntp
cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
hwclock --systohc --localtime

deterpm ntp
Decide $? "yum install ntp -y"

ntpconfig

#sysctl.conf
function sysctlconfig()
{
cat >> /etc/sysctl.conf <<EOF
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
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
net.core.netdev_max_backlog=20000
net.core.somaxconn = 2048
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.ipv4.tcp_max_syn_backlog = 16384
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
#the web behand lvs
#net.ipv4.conf.all.arp_ignore = 1
#net.ipv4.conf.all.arp_announce = 2
#net.ipv4.conf.bond0.arp_ignore = 1
#net.ipv4.conf.bond0.arp_announce = 2
EOF
}
#mem=`cat /proc/meminfo |grep MemTotal |awk '{print $2}'`
#shmax=`expr 1024 \* $mem`
#shmall=`expr $shmax / 4096`
#echo $shmax
#echo $shmall
sysctlconfig

#stop ipv6
echo "alias net-pf-10 off" >> /etc/modprobe.conf
echo "alias ipv6 off" >> /etc/modprobe.conf
/sbin/chkconfig ip6tables off

#open file limits.conf
echo "* soft nofile 65535
* hard nofile 65535
* soft nproc 65535
* hard nproc 65535" >> /etc/security/limits.conf

set +x
```

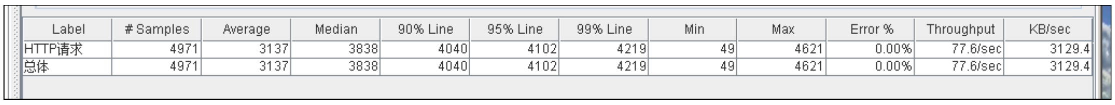
JMeter 压测结果 1

方案 1：



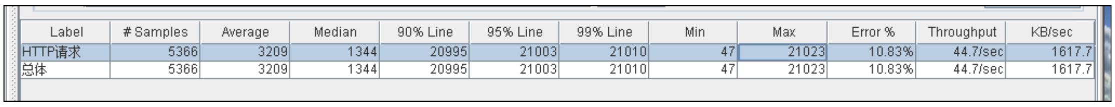
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	4852	1028	1193	1307	1365	1480	57	1977	0.00%	79.4/sec	3202.2
总体	4852	1028	1193	1307	1365	1480	57	1977	0.00%	79.4/sec	3202.2

方案 2：



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	4971	3137	3838	4040	4102	4219	49	4621	0.00%	77.6/sec	3129.4
总体	4971	3137	3838	4040	4102	4219	49	4621	0.00%	77.6/sec	3129.4

方案 3：



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	5366	3209	1344	20995	21003	21010	47	21023	10.83%	44.7/sec	1617.7
总体	5366	3209	1344	20995	21003	21010	47	21023	10.83%	44.7/sec	1617.7

nginx/jetty 调优

nginx 优化

nginx 本身已经优化得很好了，对于流量不大的站点没必要做过多的优化。

调整工作进程数

现代计算机硬件是多处理器的，NGINX 可以利用多物理或虚拟处理器。
多数情况下，你的 Web 服务器都不会配置为处理多种任务（比如作为 Web 服务器提供服务的同时也是一个打印服务器），你可以配置 NGINX 使用所有可用的处理器，NGINX 工作进程并不是多线程的。
运行以下命令可以获知你的机器有多少个处理器：
Linux 上 -
cat /proc/cpuinfo | grep processor
FreeBSD 上 -
sysctl dev .cpu | grep location
将 nginx.conf 文件中 work_processes 的值设置为机器的处理器核数。
同时，增大 worker_connections（每个处理器核心可以处理多少个连接）的值，以及将"multi_accept"设置为 ON，如果你使用的是 Linux，则也使用"epoll"：
# We have 16 cores
worker_processes 16;

```
# connections per worker
events
{
    worker_connections 4096;
    multi_accept on;
}
```

禁用访问日志文件

这一点影响较大，因为高流量站点上的日志文件涉及大量必须在所有线程之间同步的 IO 操作。

```
access_log off;
log_not_found off;
error_log /var/log/nginx-error.log warn;
```

若你不能关闭访问日志文件，至少应该使用缓冲：

```
access_log /var/log/nginx/access.log main buffer=16k;
```

启用 GZip

```
    gzip  on;
    charset UTF-8;
    gzip_disable "msie6";
    gzip_proxied any;
    gzip_min_length 1000;
    gzip_comp_level 6;
    gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml
application/xml+rss text/javascript;
```

缓存被频繁访问的文件相关的信息

```
    open_file_cache max=200000 inactive=20s;
    open_file_cache_valid 30s;
    open_file_cache_min_uses 2;
    open_file_cache_errors on;
```

调整客户端超时时间

```
client_max_body_size 500M;
client_body_buffer_size 1m;
client_body_timeout 15;
client_header_timeout 15;
keepalive_timeout 2 2;
send_timeout 15;
sendfile on;
tcp_nopush on;
tcp_nodelay on;
```

设置 **css/js/jpg/**等图片或静态文件的浏览器缓存时间

```
location ~.*\.(jpg|png|jpeg)$
{
         expires 30d;
}

location ~.*\.(js|css)?$
{
        expires 1h;
}
```

如果是静态服务器直接在 server 中加上 expires 30d;

nginx 动态缓存

http 中加入：

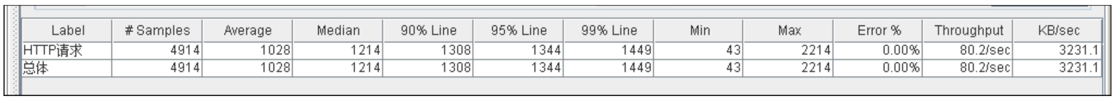
```
proxy_connect_timeout 90;
proxy_send_timeout 90;
proxy_read_timeout 90;
proxy_buffer_size 40k;
```

```
    proxy_buffers 4 320k;
    proxy_busy_buffers_size 640k;
    proxy_temp_file_write_size 640k;
    proxy_temp_path /home/goujia/project/nginxcache/temp_dir;
    proxy_cache_path /home/goujia/project/nginxcache/cache levels=1:2 keys_zone=cache_one:200m inactive=1d max_size=30g;
```

server 中加入：

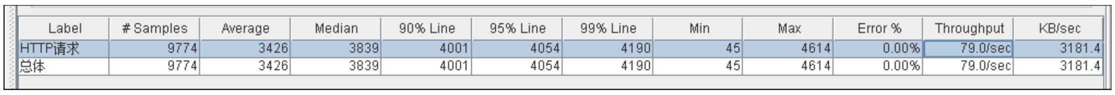
```
location / {
        index index.htm;
        proxy_cache cache_one;
        proxy_cache_valid 200 302 1h;
        proxy_cache_key $host$uri$is_args$args;
        proxy_pass     http://web;
        rewrite ^/$ /home/newIndex.htm last;
        proxy_ignore_headers "Cache-Control" "Expires" "Set-Cookie";
        expires off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
```

方案 1：



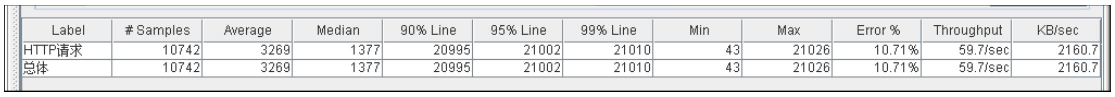
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	4914	1028	1214	1308	1344	1449	43	2214	0.00%	80.2/sec	3231.1
总体	4914	1028	1214	1308	1344	1449	43	2214	0.00%	80.2/sec	3231.1

方案 2：



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	9774	3426	3839	4001	4054	4190	45	4614	0.00%	79.0/sec	3181.4
总体	9774	3426	3839	4001	4054	4190	45	4614	0.00%	79.0/sec	3181.4

方案 3：



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	10742	3269	1377	20995	21002	21010	43	21026	10.71%	59.7/sec	2160.7
总体	10742	3269	1377	20995	21002	21010	43	21026	10.71%	59.7/sec	2160.7

启用 nginx 动态缓存的情况下，性能有一定提升。

jetty 优化

关闭 debug 日志

```
sed -i "s/debug/warn/g" log4j.xml
```

log4j.xml 文件的位置一般在 WEB-INF/classes 中

线程池优化

```
threads.min=20
threads.max=500
threads.timeout=6000
jetty.output.buffer.size=32768
jetty.request.header.size=8192
jetty.response.header.size=8192
jetty.send.server.version=true
jetty.send.date.header=false
jetty.dump.start=false
jetty.dump.stop=false
jetty.delayDispatchUntilContent=false
jsp-impl=apache
jetty.port=8080
http.timeout=3000
```

JVM 参数优化

-Xms：设置 jvm 内存的初始大小
-Xmx：设置 jvm 内存的最大值
-Xmn：设置新域的大小（这个似乎只对 jdk1.4 来说是有效的，后来就废弃了）
-Xss：设置每个线程的堆栈大小(也就是说,在相同物理内存下，减小这个值能生成更多的线程)

-XX：NewRatio :设置新域与旧域之比，如-XX：NewRatio = 4 就表示新域与旧域之比为 1：4
-XX:NewSize：设置新域的初始值
-XX:MaxNewSize ：设置新域的最大值
-XX:PermSize：设置永久域的初始值
-XX:MaxPermSize：设置永久域的最大值
-XX:SurvivorRatio=n:设置新域中 Eden 区与两个 Survivor 区的比值。（Eden 区主要是用来存放新生的对象，而两个 Survivor 区则用来存放每次垃圾回收后存活下来的对象）
监控内存 CPU
常见的错误 ：
java.lang.OutOfMemoryError 相信很多开发人员都用到过，这个主要就是 JVM 参数没有配好引起的，但是这种错误又分两种：java.lang.OutOfMemoryError: Java heap space 和 java.lang.OutOfMemoryError: PermGen space，其中前者是有关堆内存的内存溢出，可以同过配置-Xms 和-Xmx 参数来设置，而后者是有关永久域的内存溢出，可以通过配置-XX:MaxPermSize 来设置。

Content Cache

动态内容不会被 cache 静态内容才会被 cache
maxCacheSize 256,000,000
maxCachedFileSize 200,000,000
maxCachedFiles ?2,048
useFileMappedBuffer ?true
可以通过 etc/webdefault.xml 配置

冗余组件去除

去除多余的 Connector 去除不需要的构建 Handler 例如 SessionHandler,ServletHandler

代码优化

在高峰期，减去日志记录的操作，或者把日志暂时先缓存起来，使用异步处理的方式。
减少一些数据库操作。

安全检测与优化

主要分为以下几方面：

防火墙

```
# Generated by iptables-save v1.4.7 on Wed Jun 10 02:15:03 2015
*mangle
:PREROUTING ACCEPT [81:5466]
:INPUT ACCEPT [81:5466]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [57:14768]
:POSTROUTING ACCEPT [51:13720]
COMMIT
# Completed on Wed Jun 10 02:15:03 2015
# Generated by iptables-save v1.4.7 on Wed Jun 10 02:15:03 2015
*nat
:PREROUTING ACCEPT [1:234]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
#-A OUTPUT -p tcp -d 12.1.1.117/32 -j DNAT --to-destination 122.224.66.27
COMMIT
# Completed on Wed Jun 10 02:15:03 2015
# Generated by iptables-save v1.4.7 on Wed Jun 10 02:15:03 2015
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
:bad_packets - [0:0]
:bad_tcp_packets - [0:0]
:icmp_packets - [0:0]
:tcp_inbound - [0:0]
:tcp_outbound - [0:0]
:udp_inbound - [0:0]
:udp_outbound - [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -j ACCEPT
-A INPUT -j bad_packets
-A INPUT -d 224.0.0.1/32 -j DROP
-A INPUT -i eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth1 -p tcp -j tcp_inbound
-A INPUT -i eth1 -p udp -j udp_inbound
-A INPUT -i eth1 -p icmp -j icmp_packets
-A INPUT -m pkttype --pkt-type broadcast -j DROP
-A INPUT -m limit --limit 3/min --limit-burst 3 -j LOG --log-prefix "INPUT packet died: "
-A OUTPUT -p icmp -m state --state INVALID -j DROP
-A OUTPUT -s 127.0.0.1/32 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -o eth0 -j ACCEPT
-A OUTPUT -o eth1 -j ACCEPT
-A OUTPUT -m limit --limit 3/min --limit-burst 3 -j LOG --log-prefix "OUTPUT packet died: "
-A bad_packets -m state --state INVALID -j LOG --log-prefix "Invalid packet: "
-A bad_packets -m state --state INVALID -j DROP
-A bad_packets -p tcp -j bad_tcp_packets
-A bad_packets -j RETURN
-A bad_tcp_packets -p tcp -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -m state --state NEW -j LOG
--log-prefix "New not syn: "
-A bad_tcp_packets -p tcp -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -m state --state NEW -j DROP
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j LOG --log-prefix
"Stealth scan: "
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,PSH,ACK,URG
-j LOG --log-prefix "Stealth scan: "
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,PSH,ACK,URG
-j DROP
```

```
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH,URG -j LOG --log-
prefix "Stealth scan: "
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH,URG -j DROP
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,ACK,URG -j
LOG --log-prefix "Stealth scan: "
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,ACK,URG -j
DROP
-A bad_tcp_packets -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j LOG --log-prefix "Stealth scan: "
-A bad_tcp_packets -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j LOG --log-prefix "Stealth scan: "
-A bad_tcp_packets -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
-A bad_tcp_packets -p tcp -j RETURN
-A icmp_packets -p icmp -f -j LOG --log-prefix "ICMP Fragment: "
-A icmp_packets -p icmp -f -j DROP
-A icmp_packets -p icmp -m icmp --icmp-type 8 -j DROP
-A icmp_packets -p icmp -m icmp --icmp-type 11 -j ACCEPT
-A icmp_packets -p icmp -j RETURN
-A tcp_inbound -p tcp -m tcp --dport 80 -j ACCEPT
-A tcp_inbound -p tcp -m tcp -s 183.129.171.194 --dport 81 -j ACCEPT
-A tcp_inbound -p tcp -m tcp --dport 443 -j ACCEPT
-A tcp_inbound -p tcp -m tcp --dport 9011 -j ACCEPT
-A tcp_inbound -p tcp -m tcp -s 183.129.171.194 --dport 22 -j ACCEPT
-A tcp_inbound -p tcp -m tcp -s 183.129.171.194 --dport 873 -j ACCEPT
-A tcp_inbound -p tcp -j RETURN
-A tcp_outbound -p tcp -j ACCEPT
-A udp_inbound -p udp -m udp -s 112.124.49.247 -j ACCEPT
-A udp_inbound -p udp -m udp --dport 137 -j DROP
-A udp_inbound -p udp -m udp --dport 138 -j DROP
-A udp_inbound -p udp -j RETURN
-A udp_outbound -p udp -j ACCEPT
COMMIT
# Completed on Wed Jun 10 02:15:03 2015
```

简单的行为审计

记录登录用户执行的所有命令

```
[root@ay14071016231835126cz .iptables]# tail -2 /etc/profile
export HISTORY_FILE=/home/goujia/project/.usermonitor/usermonitor.log
export PROMPT_COMMAND='{ date "+%y-%m-%d %T ##### $(who am i |awk "{print \$1\" \"\
$2\" \"\$5}")  #### $(history 1 | { read x cmd; echo "$cmd"; })"; } >>$HISTORY_FILE'

[root@iZ23jo5tyxxZ .usermonitor]# cat usermonitor.sh
#!/bin/bash
upath=/home/goujia/project/.usermonitor
mkdir $upath

touch $upath/usermonitor.log

chown nobody:nobody $upath/usermonitor.log

chmod 002 $upath/usermonitor.log

chattr +a $upath/usermonitor.log

source /etc/profile
```

系统定期基础检测与通告，采用 lynis rkhunter

外部扫描 nessus