一： 介绍

通过nginx+django+uwsgi可实现负载均衡和并发请求：

nginx： Nginx是一款开源代码的高性能HTTP服务器和反向代理服务器；

django：Django是一个开放源代码的Web应用框架；

uwsgi： 可实现高并发请求；

二： 安装

1. 安装nginx

执行： rpm -ivh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

执行： yum install nginx

2. 安装django

执行： pip install django==1.9

3. 安装uwsgi

执行：yum install python-devel.x86_64

执行：pip install uwsgi

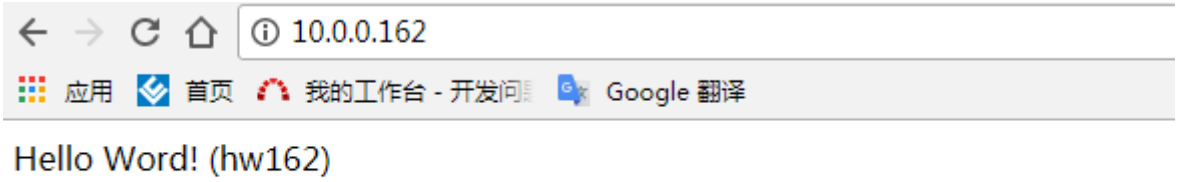
三： 准备二台虚拟机来做这个实验

10.0.0.161 web服务器

10.0.0.162 web服务器

10.0.0.162 负载均衡服务器

每个web服务节点准备一个Django工程，达到输出结果为



四： uwsgi配置

```
[root@hw162 script]# cat uwsgi.ini
[uwsgi]
# 监听IP和端口，直接做web服务器使用,作为web服务器使用
#http=0.0.0.0:9000
# 监听IP和端口，使用nginx连接时使用,10.0.0.161节点需将ip更换一下
socket=10.0.0.162:9000
# 项目根目录
chdir=/root/test01
# 服务器端程序名
wsgi-file=/root/test01/test01/wsgi.py
# 指定静态文件
#static-map=/static=/root/test01/test01/static
# 启动uwsgi的用户名和用户组
uid=root
gid=root
# 启用主进程
master=true
# 支持线程启动
enable-threads=true
# 进程个数
processes = 6
#每个进程开启4个线程
threads = 6
# 自动移除unix Socket和pid文件当服务停止的时候
vacuum=true
# 设置缓冲
buffer-size = 30000
# 进程pid文件
pidfile=/root/test01/script/uwsgi.pid
# 设置日志目录
daemonize=/root/test01/script/uwsgi.log
# 若每次请求需要花费超过该值的时间则放弃该请求，处理相应的worker被收回
harakiri=1200
```

五： nginx配置

```
[root@hw162 nginx]# cat nginx.conf
user nginx nginx;    # 定义Nginx运行的用户和用户组
worker_processes 4;  # nginx进程数，建议设置为等于CPU总核心数

error_log   /var/log/nginx/nginx.log info;  # 全局错误日志
pid         /var/run/nginx.pid;  # 进程文件

# 工作模式及连接数上线
events {
        use epoll;  # epoll模型是linux 2.6以上版本内核中的高性能网络I/O模型
        worker_connections  65535;  # 单个进程最大连接数（最大连接数=连接数*进程数）
}

#设定http服务器
http {
        include       /etc/nginx/mime.types;  #文件扩展名与文件类型映射表
        default_type  application/octet-stream;  #默认文件类型
        charset utf-8;  #默认编码

        keepalive_timeout                 120;  #长连接超时时间，单位是秒
        server_names_hash_bucket_size      128;  #服务器名字的hash表大小
        sendfile                          on;   #开启高效文件传输模式
        tcp_nopush                        on;   #防止网络阻塞
        tcp_nodelay                       on;   #防止网络阻塞
        server_tokens                     off;  # 可以关闭在错误页面中的nginx版本数字

        #用来定义记录日志的格式（可以定义多种日志格式，取不同名字即可）
        log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                          '$status $body_bytes_sent "$http_referer" '
                          '"$http_user_agent" "$http_x_forwarded_for"';


        gzip              on;      #开启gzip压缩输出
```

gzip_min_length 1k; #最小压缩文件大小

gzip_buffers 4 16k; #压缩缓冲区

gzip_http_version 1.1; #压缩版本（默认1.1)

gzip_comp_level 6; #压缩等级

#压缩类型，默认就已经包含text/html，所以下面就不用再写了，写上去也不会有问题，但是会有一个warn。

gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;

gzip_vary on; # 是否传输gzip压缩标志

gzip_disable "MSIE [1-6]\.(?!.*SV1)"; #为指定的客户端禁用gzip功能。我们设置成IE6或者更低版本以使我们的方案能够广泛兼容。

gzip_proxied any; #允许或者禁止压缩基于请求和响应的响应流。我们设置为any，意味着将会压缩所有的请求。

```
# 负载均衡
upstream test.miaohr.com {
        server 10.0.0.162:9000;
        server 10.0.0.161:9000;
}


#虚拟主机的配置
server {
        listen  80;    #监听端口
        server_name 0.0.0.0;  #监听地址
        client_max_body_size    50g;      #允许客户端请求的最大单文件字节数
        client_body_buffer_size 128k;     #缓冲区代理缓冲用户端请求的最大字节数


        #对 "/" 启用反向代理
        location / {
                include uwsgi_params;
                uwsgi_pass test.miaohr.com; # 反向代理
                uwsgi_connect_timeout 60;  # 与uwsgi-server连接的超时时间
```

```
                uwsgi_read_timeout 60;  # nginx等待uwsgi进程发送响应数据的超时时间
                uwsgi_send_timeout 60;  # nginx向uwsgi进程发送请求的超时时间
                uwsgi_param UWSGI_SCRIPT test01.wsgi; # 可有可无，入口文件，即wsgi.py相对于项目根目录的位置， “.”相当于一层目录
                uwsgi_param UWSGI_CHDIR /root/test01; # 可有可无，项目根目录

            }
            error_page 500 502 503 504 /50x.html; # 错误页
            access_log  /var/log/nginx/web.log   main;  #定义本虚拟主机的访问日志

        }

}
```

7. 负载均衡服务器

观察下列两张图片的变化：