django部署至生产环境

采用Nginx + uwsgi + django 的形式部署django WEB项目

一. 什么是uwsgi
它是一个web服务器, 也可以当做中间件使用. 如果是Nginx+uWSGI+APP形式部署项目, 那就是一个中间件; 如果是采用uWSGI+APP形式部署, 那它就是服务器.

WSGI是一个Python专有的web协议它的的发展历程是CGI >> FCGI >> WSGI >> uwsgi

CGI:
Common Gateway Interface通用网关接口, 是一个协议, 是外部应用程序(CGI应用程序)与web服务器之间的标准接口, 该协议定义了web服务器调用外部应用程序的时候需要输入的参数和给web服务器的返回结果.

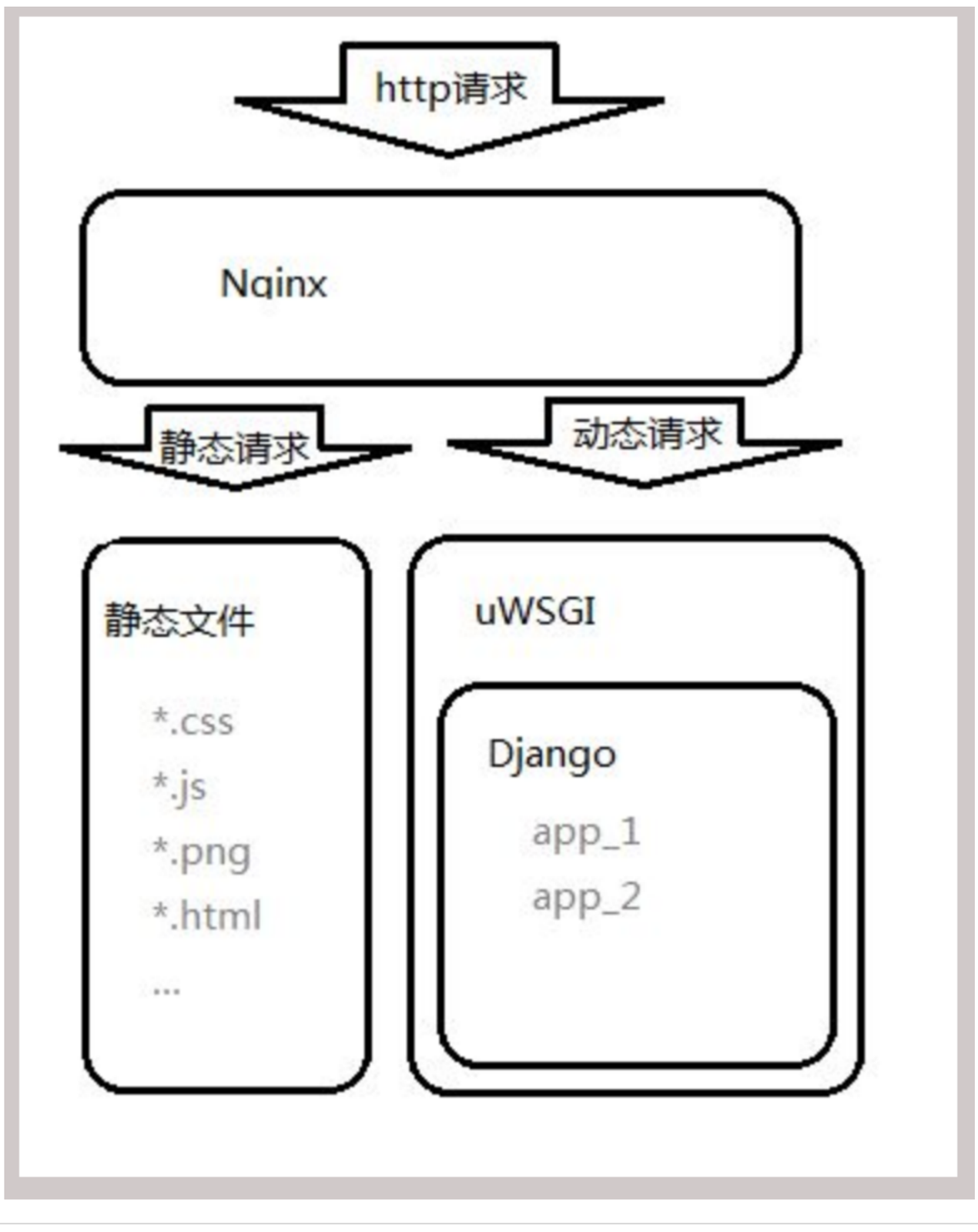
FCGI: 很明显就是比CGI更快的CGI, CGI的特点是对于一个HTTP请求就新建一个进程, 等到应用程序返回结果后就结束这个进程, 这就导致了如果有多个请求就会频繁的创建进程而产生很大的开销, 影响服务器的性能. 而FCGI就像MYSQL连接池一样, 在服务器启动的时候就新建多个空闲的进程, 当有请求的时候就分配给一个空闲的进程, 这样就避免了频繁的创建进程而导致的大开销. FCGI的另一个特点就是支持分布式, 也就是服务器和应用程序可以在不同的机器上.

WSGI: WEB SERVER GATESAY INTERFACE即web服务器网关接口, 它是Python专有的处理web服务器和应用程序的协议, 像我现在使用的Django就自带这个, 其中我们新建程序的时候有个wsgi.py就是一个WSGI兼容的web服务器的入口, 在Python的很多框架中都自带这个协议, 但是性能不好只做测试用途.

uwsgi: uWSGI是一个web服务器, 实现了WSGI协议, uwsgi协议, http协议等. 我们只需要在uwsgi的配置文件中指定application的地址, uWSGI就能直接和应用框架中的WSGI application通信. 旨在为部署分布式集群的网络应用开发一套完整的解决方案, 主要面向web及其标准服务, 由于其可扩展性, 能够被无限制的扩展用来支持更多平台和语言. uWSGI的主要特点是: 超快的性能 低内存占用 多app管理 详尽的日志功能(可以用来分析app的性能和瓶颈)高度可定制(内存大小限制, 服务一定次数后重启等)

二. Nginx在部署django项目中的作用.

Nginx是一个Http和反向代理服务器, 什么是反向代理服务器呢?
请求都会经过代理服务器处理再发给目标服务器, 这就有了使用Nginx的几个好处:
安全: 不管什么请求都要经过代理服务器, 这样就避免了外部程序直接攻击web服务器.
负载均衡: 根据请求情况和服务器负载情况, 将请求分配给不同的web服务器保证服务器性能.
提高web服务器的IO性能: 总结来说就是请求从客户端传到web服务器是需要时间的, 传递多长时间就会让这个进程阻塞多长时间, 而通过反向代理, 就可以在反向代理这完整接受请求, 然后再传给web服务器, 从而保证服务器性能, 而且有的一些简单的事情(比如静态文件)可以直接由反向代理处理, 不经过web服务器, 实现动静分离.



二. 开始部署django项目

2.1 uwsgi相关配置

2.1.1 可通过pip形式安装uwsgi也可通过下载源码包安装.

```
(webapps) [root@harbor-a tmp]# pip install uwsgi
Collecting uwsgi
  Downloading https://files.pythonhosted.org/packages/e7/1e/3dcca007f974fe4eb369bf1b8629d5e342bb3055e2001b2e5340aaefae7a/uwsgi-2.0.18.tar.gz (801kB)
    100% |████████████████████████████████| 808kB 15kB/s
Building wheels for collected packages: uwsgi
  Building wheel for uwsgi (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/2d/0c/b0/f3ba1bbce35c3766c9dac8c3d15d5431cac57e7a8c4111c268
Successfully built uwsgi
Installing collected packages: uwsgi
Successfully installed uwsgi-2.0.18
```

2.1.2 创建uwsgi配置文件

```
[uwsgi]
http = 172.16.70.231:8900
uid = root
gid = root
vacuum=false
chdir = /app_shell/python/webapps/webapps
#切换工作目录, 具体配置需与你的项目一致
home = /app_shell/python/webapps
#项目的家目录, 具体配置需与你的项目一致
wsgi-file = webapps/wsgi.py
```

```
    #指定django项目wsgi文件位置
    processes = 4
    #启动的uwsgi进程数量ps -ef | grep uwsgi
    threads = 8
    #每个进程下的线程数量, 本例中配置进程为4线程为8, 该web服务器可以提供32个并发.
    master = true
    socket=/app_shell/python/webapps/webapps/uwsgi.sock
    #定义socket文件位置
    chmod-socket=666
    #定义socket文件权限
```

2.1.3 启动uwsgi

```
    (webapps) [root@harbor-a webapps]# pwd
    /app_shell/python/webapps/webapps
    (webapps) [root@harbor-a webapps]# uwsgi  --ini ./uwsgi.ini

        *** Starting uWSGI 2.0.18 (64bit) on [Sat Feb  2 04:34:56 2019] ***
    compiled with version: 4.8.5 20150623 (Red Hat 4.8.5-36) on 02 February 2019 09:23:12
    os: Linux-3.10.0-327.el7.x86_64 #1 SMP Thu Nov 19 22:10:57 UTC 2015
    nodename: harbor-a
    machine: x86_64
    clock source: unix
    pcre jit disabled
    detected number of CPU cores: 1
    current working directory: /app_shell/python/webapps/webapps
    detected binary path: /app_shell/python/webapps/bin/uwsgi
    *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
    chdir() to /app_shell/python/webapps/webapps
    your processes number limit is 3828
    your memory page size is 4096 bytes
    detected max file descriptor number: 1024
    lock engine: pthread robust mutexes
    thunder lock: disabled (you can enable it with --thunder-lock)
    uWSGI http bound on 172.16.70.231:8900 fd 3
    uwsgi socket 0 bound to UNIX address /app_shell/python/webapps/webapps/uwsgi.sock fd 6
    *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
    Python version: 3.6.1 (default, Jan 30 2019, 12:05:36)  [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]
    Set PythonHome to /app_shell/python/webapps
    Python main interpreter initialized at 0x18d86d0
    *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
    python threads support enabled
    your server socket listen backlog is limited to 100 connections
    your mercy for graceful operations on workers is 60 seconds
    mapped 730560 bytes (713 KB) for 32 cores
    *** Operational MODE: preforking+threaded ***
    WSGI app 0 (mountpoint='') ready in 1 seconds on interpreter 0x18d86d0 pid: 17826 (default app)
    *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
    *** uWSGI is running in multiple interpreter mode ***
    spawned uWSGI master process (pid: 17826)
    spawned uWSGI worker 1 (pid: 17828, cores: 8)
    spawned uWSGI worker 2 (pid: 17829, cores: 8)
    spawned uWSGI worker 3 (pid: 17830, cores: 8)
    spawned uWSGI worker 4 (pid: 17831, cores: 8)
    spawned uWSGI http 1 (pid: 17832)


    ###综上 uwsgi启动成功###
```

2.2 Nginx相关配置

2.2.1 安装Nginx

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

    rpm -Uvh nginx-release-centos-7-0.el7.ngx.noarch.rpm


    yum -y install nginx
```

2.2.2 配置Nginx, 修改nginx配置文件

```
    (webapps) [root@harbor-a nginx]# vim /etc/nginx/nginx.conf

    server {
    listen       8080;
    server_name  localhost;
    location / {
        include uwsgi_params;
        uwsgi_pass unix:/app_shell/python/webapps/webapps/uwsgi.sock;
    }
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
       }
    }
```

2.2.3 重启nginx服务

```
      systemctl restart nginx
```

2.3 测试访问

```
    (webapps) [root@harbor-a nginx]#  netstat -tuan | grep 8080
    tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN

    (webapps) [root@harbor-a nginx]# netstat -tuan | grep 8900
    tcp        0      0 172.16.70.231:8900      0.0.0.0:*               LISTEN
```

hello 刘硕的教程
this is Python学习手册
this is Python核心编程
this is Python CookBook

your name is liushuo and your age is 20
不及格 欢迎你 172.16.70.1

用户--->nginx:8080 --> uwsgi:8900 --> django app, django项目部署成功!