自己动手做一台 SLAM 导航机器人

附录 B：用于 ROS 机器人管理调度的后台服务器搭建

作者：

知乎@小虎哥哥爱学习

目录

centos7 下部署 Django 后台服务器用于 ROS 机器人管理调度：

- nginx
- uwsgi
- django
- python3

0. 安装步骤预览

(1)系统默认自带 python2.x，所以需要先安装 python3.x

(2)python2 对应 pip，python3 对应 pip3，用源码安装 python3 后 pip3 也自动安装了

(3)用 pip3 安装 virtualenv

(4)用 virtualenv 创建 python3 的虚拟环境

(5)在创建的虚拟环境中，用 pip3 安装 Django 和 uwsgi

(6)安装 nginx

(7)创建 django 项目

(8)关联 nginx、uwsgi、django

注：使用 root 身份登录系统执行

1. 编译安装 python3

（1）依赖安装

```
#为 centos 系统增加编译功能：

yum -y install gcc gcc-c++



#防止编译安装 python3 出现各种异常：

yum install wget openssl-devel bzip2-devel expat-devel gdbm-devel readline-devel
sqlite-devel
```

（2）编译安装

```
#下载 python3 安装包：
```

```
cd /home/<username>/Downloads/ #<username>用自己的用户名代替

wget https://www.python.org/ftp/python/3.6.3/Python-3.6.3.tgz



#解压:

tar -zxvf Python-3.6.3.tgz



#配置,将 python3 安装到/usr/local/python3/路径下:

cd Python-3.6.3

./configure --prefix=/usr/local/python3



#编译安装:

make -j2

make install -j2



#建立软链接，方便在终端中直接使用 python3 和 pip3 命令:

ln -s /usr/local/python3/bin/python3.6 /usr/bin/python3

ln -s /usr/local/python3/bin/pip3 /usr/bin/pip3



#安装成功性测试,显示相应版本就表示成功了:

python3 -V

pip3 -V
```

2.用 pip3 安装 virtualenv

```
#更新 pip3 至最新版本

pip3 install --upgrade pip



#安装 virtualenv
```

```
pip3 install virtualenv



#赋予可执行权限

chmod 777 /usr/local/python3/lib/python3.6/site-packages/virtualenv.py

#建立软链接，方便在终端中直接使用 virtualenv

ln -s /usr/local/python3/lib/python3.6/site-packages/virtualenv.py
/usr/bin/virtualenv
```

3.用 virtualenv 创建 python3 的虚拟环境

```
#切换到用户家目录下

cd /home/<username>/



#新建文件夹

mkdir www_space_venv

cd www_space_venv



#创建名称为 venv 的虚拟环境

virtualenv -p /usr/bin/python3 venv
```

4.在创建的虚拟环境中，用 pip3 安装 django 和 uwsgi

```
cd /home/<username>/www_space_venv/



#激活虚拟环境

source venv/bin/activate



#安装 django 与 uwsgi

pip3 install django
```

```
pip3 install uwsgi



#查看 django 安装成功与否

pip3 show django

#查看 uwsgi 安装成功与否

pip3 show uwsgi



#退出虚拟环境

deactivate
```

5. 安装 nginx

```
#添加 nginx 存储库

yum install epel-release



#安装 nginx

yum install nginx



#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：这个我也没看懂，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
```

6. 创建一个 django 的项目

（1）新建项目

```
#切换到工作目录
```

```
cd /home/<username>/www_space_venv/



#激活虚拟环境

source venv/bin/activate



#创建自己的 django 项目，项目取名为 mysite_django

django-admin.py startproject mysite_django



#修改 settings.py，允许所有 HOST 的访问，不然浏览器访问会报错 DisallowedHost at / Invalid HTTP_HOST header

ALLOWED_HOSTS = [ ] #修改前

ALLOWED_HOSTS = ['*'] #修改后



#启动 django 自带 web 服务器

cd mysite_django

python3 manage.py runserver <自己主机地址 IP>:8080 #<自己主机地址 IP>填自己主机地址，后面需要指定一个可用的端口(如 8080)



#用浏览器访问 django,<自己主机地址 IP>:8080，得到如下消息说明成功

The install worked successfully! Congratulations!
```

（2）新建应用

```
#切换到工作目录

cd /home/<username>/www_space_venv/



#激活虚拟环境

source venv/bin/activate
```

```
#新建应用

cd mysite_django

python3 manage.py startapp hello_app



#新定义的应用加到 settings.py 中的 INSTALL_APPS 中

INSTALLED_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'hello_app',

]
```

编辑应用中的 views.py 文件：

```
from django.http.response import HttpResponse



def hello(request):

    user = request.GET.get('user')

    if not user: user = 'world'

    return HttpResponse('hello %s' % user)
```

编辑项目中的 urls.py 文件：

```
from django.contrib import admin

from django.urls import path



from hello_app import views as hello_views
```

```
urlpatterns = [
path('admin/', admin.site.urls),
path('hello',hello_views.hello),
]
```

（3）测试

```
#启动 django 自带 web 服务器
cd mysite_django
python3 manage.py runserver <自己主机地址 IP>:8080 #<自己主机地址 IP>填自己主机地址，后面需要指定一个可用的端口(如 8080)

#用浏览器访问 django,访问方式为
<自己主机地址 IP>:8080/hello
<自己主机地址 IP>:8080/hello?user=123

#浏览器可以看到对应返回信息，说明 django 项目新建成功
```

7.关联 nginx、uwsgi、django

（1）防火墙中相应端口开放，允许外网访问

默认开放给外网 http 访问的端口是 80，所以需要在服务器的防火墙中允许 80 端口，不然外网的请求进不了服务器；

如果想开放给外网 http 访问的端口是其他端口（如 8080），依照下面的例子，在服务器的防火墙中允许该端口（8080），同时用步骤（2）方法开放允许 http 访问的端口（8080）。

```
#以 80 端口为例

#查询 TCP/UDP 的 80 端口占用情况
firewall-cmd --query-port=80/tcp
```

```
firewall-cmd --query-port=80/udp

#如果返回结果为“no”，则表示该端口尚未开放



#永久开放 TCP/UDP 的 80 端口

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

firewall-cmd --permanent --zone=public --add-port=80/udp



#重启防火墙

firewall-cmd --reload
```

（2）开放允许 http 访问的端口

确保外网访问 nginx 服务器的 http 端口、nginx 与 uwsgi 通信的 socket 的端口都在 http 访问端口的列表中。

```
#查看 http 允许访问的端口

semanage port -l | grep http_port_t

#输出结果

http_port_t                    tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000



#将需要开放的端口加入到如上端口列表中，例如开放 7878 端口作为 nginx 与 uwsgi 通信的 socket 通道

semanage port -a -t http_port_t  -p tcp 7878
```

（3）关联 nginx 与 uwsgi

#为 django 站点创建一个 nginx 服务的配置文件

```
cd /etc/nginx/conf.d/

touch mysite_django.conf

vim mysite_django.conf
```

#在文件 mysite_django.conf 中填入如下内容

```
server {
```

```
    server_name 192.168.2.141; #暴露给外部访问的 IP 地址，根据实际情况改写成自己主机 IP

    listen 80; #暴露给外部访问的端口，确保端口在 http 访问和防火墙访问的允许列表中

    location / {

        include uwsgi_params;

        uwsgi_pass 127.0.0.1:7878; #nginx 与 uwsgi 通信用的 socket 接口，确保端口在 http 访问端口的列表中

    }

}
```

#如遇到 nginx 服务启动失败，请检查 mysite_django.conf 中指定的端口是否被占用

（4）关联 uwsgi 与 django

#创建 uwsgi 配置文件

```
cd /home/<username>/www_space_venv

touch mysite_django_uwsgi.ini

vim mysite_django_uwsgi.ini
```

#在 mysite_django_uwsgi.ini 文件中填入如下内容

```
#mysite_django_uwsgi.ini

[uwsgi]



#与 nginx 通信

socket = 127.0.0.1:7878

#让 uwsgi 作为单独的 web-server,这里注释掉

#http = 127.0.0.1:7878



#django 项目根目录

chdir = /home/chatbot/www_space_venv/mysite_django #根据实际情况改写成自己 django 项目的路径
```

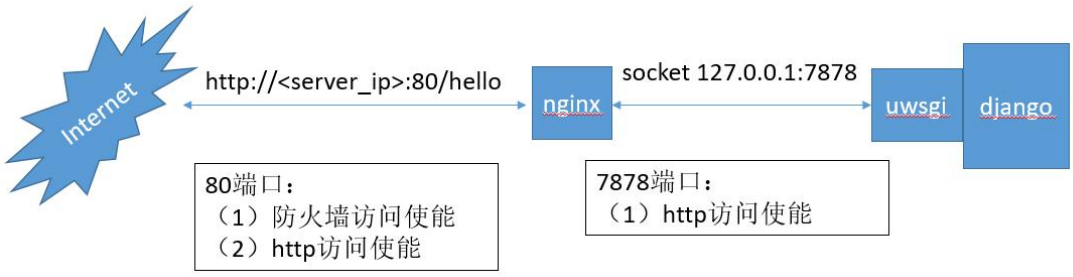
```
#本项指示 uwsgi.py 文件的位置，其位于 Django 工程目录下有个与工程名同名的子文件夹内(设置方式为:文件夹名.wsgi)

#module = mysite_django

wsgi-file = mysite_django/wsgi.py


processes = 4

threads = 2

master = True

pidfile = uwsgi.pid

daemonize = uwsgi.log


# 虚拟环境地址

#virtualenv = /home/chatbot/www_space_venv/venv
```

（5）nginx+uwsgi+django 联调测试

```
#先关闭 nginx 与 uwsgi 服务

pkill -9 nginx

pkill -9 uwsgi


#启动 nginx 服务

service nginx start


#启动 uwsgi 服务

cd /home/chatbot/www_space_venv

source venv/bin/activate #激活虚拟环境

uwsgi --ini mysite_django_uwsgi.ini
```

```
#用浏览器访问 django,访问方式为

<自己主机地址 IP>:80/hello

<自己主机地址 IP>:80/hello?user=123



#浏览器可以看到对应返回信息，说明部署成功，后面只需专注于 django 项目的开发了
```

联调架构如下图所示：



后记

为了防止后续大家找不到本篇文章，我同步制作了一份文章的 pdf 和本专栏涉及的例程代码放在 github 和 gitee 方便大家下载，如果下面给出的 github 下载链接打不开，可以尝试 gitee 下载链接：

- github 下载链接：
  https://github.com/xiihoo/DIY_A_SLAM_Navigation_Robot

- gitee 下载链接：
  https://gitee.com/xiihoo-robot/DIY_A_SLAM_Navigation_Robot

参考文献

张虎,机器人 SLAM 导航核心技术与实战[M].机械工业出版社,2022.