python celery 异步任务队列

博客：http://www.linuxyw.com
作者：戴儒锋
邮箱：63780668@qq.com

celery 简介：

celery 是一个异步任务队列/基于分布式消息传递的作业队列。它侧重于实时操作，但对调度支持也很好。 celery 是用 Python 编写的，但该协议可以在任何语言实现。更多简介的请自己在网上搜索

本文目的是用 python 使用 celery 做异步任务，在 centos 6.4 上安装 celery，并用 supervisor 来管理 celery 进程，celery 采用 redis 做中间件的消息传输。现实中可以 celery 做异步请求，如发送邮箱，发送消息，请求 URL 等场景

环境部署

环境

系统：centos 6.4 64 位
python 版本：2.6.6

创建相关的目录

```
mkdir -p /data/{redis,logs,www}
```

（我会把 python 脚本都放在/data/www 中）

安装 celery

```
yum install python-devel python-setuptools
easy_install pip
pip install celery
pip install redis     #（python 中的 redis 模块）
```

安装 supervisor

```
pip install supervisor
echo_supervisord_conf > /etc/supervisord.conf
```

如果在执行 echo_supervisord_conf > /etc/supervisord.conf 时报 pkg_resources.DistributionNotFound: meld3>=0.6.5 错误的话，找到 supervisor-3.1.3-py2.6.egg-info/requires.txt，把文件里面 meld3 >= 0.6.5 注释掉，然后再执行

echo_supervisord_conf > /etc/supervisord.conf 就好了

查找方法：

```
find / | grep requires.txt
```

配置

```
vim /etc/supervisord.conf
```

在配置后面添加以下参数

```
[program:celery]
command=/usr/bin/celery worker -A tasks
directory=/data/www
stdout_logfile=/data/logs/celery.log
autostart=true
autorestart=true
redirect_stderr=true
stopsignal=QUIT
```

注释

;program:celery 要管理的进程名，你自己随便定义，我这定义了叫 celery
;command 是启动 celery 的命令
;directory 是程序目录 ，因为我要启动 celery，需要进入/data/www 目录中才能生效的，所以这里在启动命令时，会切换到这个目录里
;autorstart 自动重启 celery
;stdout_logfile 存放 celery 的日志路径

以上命令的大概意思就是：
进行到/data/www 目录，然后执行/usr/bin/celery worker -A tasks，并把输出的日志保存到/data/logs/celery.log 中，这是指定了 worker 模式，如果不指定，默认为 prefork 模式，一般你机器有几核，系统就开启几个 worker 进程，如果有异常，记得查看日志/data/logs/celery.log

安装 redis

```
cd /usr/local/src
wget http://download.redis.io/releases/redis-3.0.5.tar.gz
tar xf redis-3.0.5.tar.gz
cd redis-3.0.5
make
make install
```

(可用 make PREFIX=/usr/local/redis install 安装到指定的路径下面)

```
cp utils/redis_init_script /etc/init.d/redis
chmod a+x /etc/init.d/redis
```

```
mkdir /etc/redis
cp redis.conf /etc/redis/
```

简单修改 redis.conf（前面数字是行号）

```
 42 daemonize yes                                  #进程转入后台运行
 50 port 22222                                     #修改端口，不用默认的 6379
 69 bind 127.0.0.1                                 #绑定 IP，只能本地连接
192 dir /data/redis                                #修改 redis 文件存放路径
396 requirepass DILSi3_dflka_f3i_LFKDkdf!idf       #设置密码
453 maxmemory 256M        #redis 最大使用 256M 内存，我的是虚拟机，所以内存设置小
```

注释：

requirepass DILSi3_dflka_f3i_LFKDkdf!idf　是配置 redis 密码，客户端连接和关闭 redis 时需要此密码

修改/etc/init.d/redis（前面数字是行号）

```
  6 REDISPORT=22222
  7 EXEC=/usr/local/bin/redis-server
  8 CLIEXEC=/usr/local/bin/redis-cli
  9 PASSWD="DILSi3_dflka_f3i_LFKDkdf!idf"    #增加行
 10 PIDFILE=/var/run/redis.pid
 11 CONF="/etc/redis/redis.conf"
 30                 $CLIEXEC -p $REDISPORT -a $PASSWD    shutdown
```

注释：

关闭 redis 时，需要带上 redis 密码，否则失败，如果没有设置密码，则不需要带密码关闭

启动 redis

service redis start

查看 redis 进程

ps -ef |grep redis

root　　28631　　1　0 04:30 ?　　00:00:00 /usr/local/bin/redis-server 127.0.0.1:22222

root　　28635　1542　0 04:31 pts/0　　00:00:00 grep redis

编写 celery 脚本、

```
cd /data/www
vim tasks.py
```

```
#!/usr/bin/env python
```

```
#coding:utf-8

import time

#导入 celery 相关模块、方法
from celery import Celery
from celery import platforms

#因为 supervisord 默认是用 root 运行的，必须设置以下参数为 True 才能允许 celery 用 root 运行
platforms.C_FORCE_ROOT = True
#配置 celery，连接 redis，DILSi3_dflka_f3i_LFKDkdf!idf 是 redis 的密码，22222 是端口 10 是库（redis 有 16 个库，这取第 11 个，即 10）
config={}
config['CELERY_BROKER_URL'] = 'redis://:DILSi3_dflka_f3i_LFKDkdf!idf@127.0.0.1:22222/10'
config['CELERY_RESULT_BACKEND'] = 'redis://:DILSi3_dflka_f3i_LFKDkdf!idf@127.0.0.1:22222/10'
#不需要返回任务状态，即设置以下参数为 True
config['CELERY_IGNORE_RESULT'] = True

app = Celery("tasks", broker=config['CELERY_BROKER_URL'])
app.conf.update(config)


@app.task
def output(num):
    """输出传进来的数字，在输出前等待 2 秒"""
    time.sleep(2)
    print num
```

启动 supervisord

```
/usr/bin/supervisord
```

查看是否启动成功

```
[root@drfdai www]# ps -ef |grep supervisor
root      28656      1  0 04:52 ?        00:00:00 /usr/bin/python /usr/bin/supervisord
root      28692   1542  0 04:53 pts/0    00:00:00 grep supervisor
```

启动 celery：

```
supervisorctl start celery
```

查看 celery 是否连接成功 redis

```
tailf /data/logs/celery.log
```

能看到[2015-11-16 05:05:27,938: WARNING/MainProcess] celery@drfdai ready.说明连接成功

测试

在终端【终端 1】中打开实时日志：

```
[root@drfdai www]# tailf /data/logs/celery.log
```

打开另一终端【终端 2】测试：

```
[root@drfdai ~]# cd /data/www
[root@drfdai www]# python
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from tasks import output
>>> for i in xrange(20):
...         output.apply_async(args=[i])
...
<AsyncResult: 38a14673-3ce6-4244-b262-d488aa637ff9>
<AsyncResult: 3ff1b2c0-4b37-4c70-a543-d7c58111cb99>
<AsyncResult: 355c232f-c419-43d5-bf8f-afa4d7084136>
<AsyncResult: c3089d05-d430-4fd4-bfba-112d6dd5b635>
<AsyncResult: 842db1ad-5490-456d-82d8-e81e60db244b>
<AsyncResult: fe0dd54a-8bad-43e2-a8b8-b87be158cf16>
<AsyncResult: edd95275-a3b3-46a5-8e21-b88c7faedbe9>
<AsyncResult: c21b9c62-644d-4171-bb68-9a371af60283>
<AsyncResult: 29d38896-9d4d-4b97-8e27-6a7209f06dc9>
<AsyncResult: 7c3ba209-01fd-44d3-9645-f9c85d799eaf>
<AsyncResult: b08bbb7a-341b-436c-8ef4-be954b1fcbc5>
<AsyncResult: b03193a2-c6d6-49ff-ac27-b453593050c4>
<AsyncResult: 1be6394c-09c3-4f48-9ed4-1114fd1af637>
<AsyncResult: b496cdd3-3032-41ad-939e-414198c9056a>
<AsyncResult: 0dd365ea-383b-4c93-9ed0-406bcff6f41b>
<AsyncResult: ce60892e-d640-4589-8a8b-3a7f18e8c6d3>
<AsyncResult: df4fd3c4-d9e8-474b-b60f-c6e11283564a>
<AsyncResult: 83e7d36c-fc42-4f3b-8941-df40fa13a517>
<AsyncResult: 9b25f7f7-c3bc-4bb0-8a1f-27ceec59cf0c>
<AsyncResult: f0d7e201-032b-41a0-8e7a-c02681c31947>
```

切换到【终端 1】可以看到，每隔 2 秒就会打印一行日志

```
   warnings.warn(CDeprecationWarning(W_PICKLE_DEPRECATED))
[2015-11-16 05:58:09,687: WARNING/MainProcess] celery@drfdai ready.
[2015-11-16 05:59:18,816: WARNING/Worker-1] 0
```

```
[2015-11-16 05:59:19,826: WARNING/Worker-1] 1
[2015-11-16 05:59:20,830: WARNING/Worker-1] 2
[2015-11-16 05:59:21,836: WARNING/Worker-1] 3
[2015-11-16 05:59:22,841: WARNING/Worker-1] 4
[2015-11-16 05:59:23,848: WARNING/Worker-1] 5
[2015-11-16 05:59:24,853: WARNING/Worker-1] 6
[2015-11-16 05:59:25,858: WARNING/Worker-1] 7
[2015-11-16 05:59:26,864: WARNING/Worker-1] 8
[2015-11-16 05:59:27,873: WARNING/Worker-1] 9
[2015-11-16 05:59:28,884: WARNING/Worker-1] 10
[2015-11-16 05:59:29,892: WARNING/Worker-1] 11
[2015-11-16 05:59:30,899: WARNING/Worker-1] 12
[2015-11-16 05:59:31,908: WARNING/Worker-1] 13
[2015-11-16 05:59:32,912: WARNING/Worker-1] 14
[2015-11-16 05:59:33,920: WARNING/Worker-1] 15
[2015-11-16 05:59:34,926: WARNING/Worker-1] 16
[2015-11-16 05:59:35,933: WARNING/Worker-1] 17
[2015-11-16 05:59:36,939: WARNING/Worker-1] 18
[2015-11-16 05:59:37,944: WARNING/Worker-1] 19
```

当我们在【终端 2】中调用此函数后，执行并没有受到 time.sleep(2)的影响，执行后马上就结束了，而【终端 1】每隔 2 秒就输出 1 行日志，这日志是函数 output 输出来的。

实用代码片断：

vim common/tasks.py

```
#导入发送邮件模块
from email.mime.text import MIMEText
import smtplib
import json
import datetime

from celery import Celery
from celery import platforms
import requests

#导入配置文件
from config import setting

#发送邮箱配置
smtp = setting.smtp
user = setting.user
```

```
passwd = setting.passwd

此处省略……

@celery.task
def sendMail(mail):
    '''
    to_list 接收者邮件，每个邮件地址用","分隔，str 格式
    subuect 邮件主题，str 格式
    context 邮件内容，str 格式
    '''
    to_list = mail.get('to_list')
    subject = mail.get('subject')
    context = mail.get('context')
    msg = MIMEText(context,'html','utf-8')
    msg['Subject'] = subject
    msg['From'] = "linux 系统运维<%s>" % user
    to_list = list(to_list.split(','))
    msg['To'] = ",".join((to_list))
    try:
        s = smtplib.SMTP()
        s.connect(smtp)
        s.login(user,passwd)
        s.sendmail(user,to_list,msg.as_string())
        s.close()
        return json.dumps({"redid":0,"redmsg":"发送成功"})
    except Exception,e:
        print 'Error: %s' % e
        return json.dumps({"redid":-1,"redmsg":"发送失败"})
```

在实际项目中调用此方法，做异步邮箱发送，发送后不需要等待返回的结果

```
#/usr/bin/env python
#coding:utf-8
from common.tasks import sendMail
sendMail.delay(dict(to_list='xxxxxx@qq.com',subject='测试邮件',context='测试'))
```