Sistemas Distribuidos

Practica de Colas

Escuela superior de Ingeníeria
Universidad de Cádiz

Tema

En nuestro trabajo hemos decidido realizar una búsqueda de una palabra clave, en este caso en Twitter, y subir a Dropbox lo que actualmente se esta hablando sobre dicha palabra en la red social.

Descripción

Para realizar esto hemos seguido una serie de pasos:

1. Hemos afrontado el problema desde varias perspectivas y lenguajes, finalmente hemos decidido realizar el codigo principal en Python ya que es el lenguaje que te ofrece mas facilidades a la hora de utilizar las librerias de RabittMQ y Celery ademas de la utilización de la libreria de pika que es una libreria de cliente para Python.

2. Hemos realizado un código para el cliente y otro para el servidor, de esta forma el cliente le pasaría la palabra que quiere buscar al servidor ,que seria la peticion al servidor, con dichas peticiones el servidor recogería los tweets que contienen la palabra clave.

3. Una vez recogidos los tweets seran devueltos al cliente y tratados, el cual escribiría en un documento que se subira a una nube, en este caso Dropbox. Con estos datos el cliente realiza una grafica que se explicara mas adelante.

Implementación

Código del Cliente:

```
1
2  import tasks
3  import dropbox
4  from dropbox.files import WriteMode
5  import tempfile
6  import re
7  import matplotlib.pyplot as plt
8  import numpy as np
9
10 import matplotlib.image as mpimg
11 cadena = input()
12
13
14 searched_tweets = tasks.add.delay(cadena)
15
16 search=searched_tweets.get()
17
18 token="hS7FuAKGNwAAAAAAAAACbjtftARV8IHKakgKHf52Tak2nyoLMFUsAQ0skoHKpGql"
19 dbx=dropbox.Dropbox(token)
20
21 dbx.files_download_to_file("tweets.txt", '/resubida.txt')
```

```
22 f=open("tweets.txt","a")
23 f.write(cadena+" <type2:>")
24 for tweet in search:
25                 print(tweet.text)
26 
27                 f.write(tweet.text)
28 
29 f.write(" <type:> ")
30 dbx.files_delete('/resubida.txt')
31 
32 
33 f.close()
34 f=open("tweets.txt","rb")
35 with open("tras.txt","wb") as ef:
36         data=f.read()
37         ef.write(data)
38 fname="/resubida.txt"
39 #dbx.files_upload(data,fname, mute=True)
40 dbx.files_upload(data, '/resubida.txt', mode=dropbox.files.WriteMode.
      overwrite)
41 f.close()
42 f=open("tweets.txt","r")
43 fig=plt.figure()
44 
45 frafinal=f.read()
46 
47 frasefin=frafinal.split("<type:>")
48 
49 fraseskfin=[]
50 frasesti=[]
51 nombres=[]
52 tam =[]
53 print(frasefin)
54 frasefin.remove(' ')
55 for i in frasefin:
56     frasesk=i.split("<type2:>")
57     tim=len(frasesk[1])
58     tam.append(tim)
59     frasesti.append(frasesk[1])
60     nombres.append(frasesk[0])
61     fraseskfin.append(frasesk)
62 
63 totalt=len(tam)
64 y=[]
65 for i in range(1,totalt+1):
66     y.append(i)
67 
68 mos=tam
69 #data
```

```
70 names=[]
71 for i in range(0,totalt):
72     names.append(nombres[i])
73 
74 ax = plt.subplot()
75 width=0.5
76 bins = y
77 ax.bar(bins,mos,width=width)
78 ax.set_xticks(bins)
79 ax.set_xticklabels(names,rotation=45, rotation_mode="anchor", ha="right"
       )
80 
81 fig.savefig('GraficaL.png')
82 
83 f=open('GraficaL.png',"rb").read()
84 
85 dbx.files_upload(f,'/GraficaL.png', mode=dropbox.files.WriteMode.
       overwrite)
```

En el código del cliente se ve como utilizamos la API de Dropbox para descargar y subir archivos, en concreto el documento con los tweets y la gráfica. Para hacer la gráfica cogemos el documento,lo actualizamos, lo tratamos, sacamos los datos necesarios para hacer la gráfica y lo volvemos a subir a Dropbox. Una vez obtenidos los datos realizamos la gráfica que sera actualizada en Dropbox.

Código del Servidor:

```
87 
88 from celery import Celery, task
89 import tweepy
90 # Crea usando RPC para devolver los datos, y el broker AMQP
91 app = Celery("tasks", backend="rpc://",broker="pyamqp://guest@localhost/
       /")
92 # Con no_ack no espera un ack al terminar
93 @app.task(no_ack=True)
94 
95 def add(palabraclave):
96         #consumer_key ="zdcT9wn20gr1xArbSwnzcgjnp"
97         consumer_key ="D7okdzSH7aXMDT5KpWi5Lz0XM"
98 
99         consumer_secret="
              WJSYHDr657fu4x0dKswNyIDHUwSJNdtUWnFHXZOEK6daJpCuXD"
100        #consumer_secret="
              sRk7S1oHQX9GFLWMQF4BDcUYotx8gDrtUnYSYbl3du4FtjYlos"
101 
102        #access_token="978281983586525189-
              0P22hP32jO8YUkZBwXKr96MH6ozynKU"
103        access_token="981946434995081216-PtRustI3iCZxQxoHg7JzfjH2wYdTCTZ
              "
104        #access_token_secret ="
              j6mKdKFcSST4acXw6vDmF18PC10IMgcu0pA0uUNN1vYgq"
```

```
105         access_token_secret ="
               cN3ZmQqyKavZ3DNNXRhyu6goKqIW7K4gzPK5szc1XLZr6"
106
107         auth= tweepy.OAuthHandler(consumer_key,consumer_secret)
108         auth.set_access_token(access_token,access_token_secret)
109         api= tweepy.API(auth)
110
111         public_tweets=api.home_timeline();
112         tweets=api.user_timeline()
113         query =palabraclave
114         max_tweets = 5
115
116         searched_tweets = [status for status in tweepy.Cursor(api.search
               , q=query,lang="es").items(max_tweets)]
117         for tweet in searched_tweets:
118                 print(tweet.text)
119         return searched_tweets
```

En el código del Servidor es donde obtenemos los tweets, metemos las credenciales, buscamos la palabra clave y devolvemos los tweets.

Todo el envio y recogida de datos por colas esta controlado por Celery a partir de RabittMQ.

Como se ve en el codigo hemos utilizado las librerias y métodos de RabittMQ, Celery y sus metodos, finalmente decidimos no utilizar pika ya que no nos era necesario para lo que queriamos hacer.

Tecnologías utilizadas

Las tecnologías que hemos considerado utiles y necesarias para realizar el proyecto son:

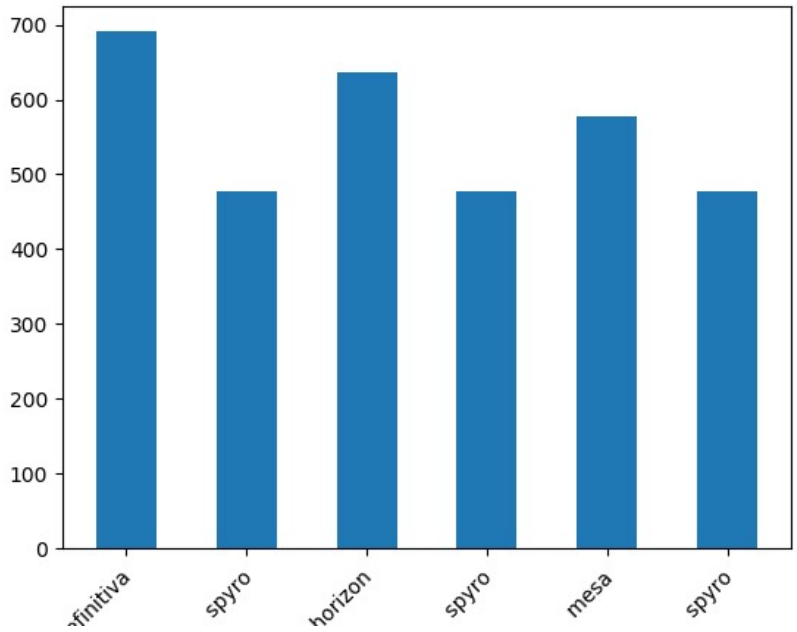
1. RabittMQ: Es un software de negociación de mensajes de código abierto que entra dentro de la categoría de middleware de mensajería. Esta tecnología es una parte esencial de nuestro proyecto dado que la vamos a utilizar como nucleo a la hora de hacer las peticiones desde la parte del cliente y envio desde la parte del servidor, además de encargarse de las colas de mensaje.

2. Celery es un sistema distribuido simple, flexible y confiable para procesar grandes cantidades de mensajes, además que proporciona a las operaciones las herramientas necesarias para mantener dicho sistema. Es una cola de tareas que se enfoca en el procesamiento en tiempo real y, al mismo tiempo, admite la programación de tareas. Con esta tecnología hemos realizado el control de colas que vienen de RabittMQ para el control asíncrono de las peticiones.

3. Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y con tabletas y móviles. Esta teconología la hemos utilizado como base de datos de los tweets. En dicha nube hemos ido subiendo los tweets que pedia el cliente y la gráfica.

Gráfica

Lo que se muestra a continuación es un ejemplo de la gráfica subida a Dropbox:



Como podemos observar en la gráfica, en el eje de las X se muestran las palabras clave y en el eje Y la frecuencia u longitud de los tweets para poder hacer una comparativa.

Modelo final utilizado y Conclusión

Para resumir, el modelo que hemos seguido es un modelo compuesto por una parte del cliente que se encarga de mandar la petición sobre la palabra clave escrita al servidor, la parte del servidor que se encarga de recibir la palabra clave y buscar en Twitter los tweets que contengan dicha palabra, dichos tweets se devolveran a la parte del cliente el cual realizara una serie de procesos, todo eso a traves de unas colas de mensajes realizadas por RabittMQ y controladas por Celery de manera asincrona. El resultado final sería una subida a Dropbox de los tweets y la gráfica realizado desde la parte del cliente con los datos, en este caso tweets, que ha devuelto el servidor, esto nos daría una idea de en que temas se escriben tweets mas largos y en cuales no se escribe apenas nada.