ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

Distributed Systems

CloudButton: Serverless Data Analytics

Autors:
Ayman Bourramouss
Usama Benabdelkrim Zakan
Grau:
Grau d'Enginyeria Informàtica
Professors:
Pedro Garcia Lopez
Josep Sampe Domenech
Data:
20/06/2021

Introducció

Aquesta pràctica es presenta com a segona i última pràctica de l'assignatura de Sistemes Distribuïts.

En primer lloc, la pràctica consisteix en programar una aplicació Big Data. Això, implica la cerca de les dades que s'usen per satisfer la idea principal del projecte. Aquest primer pas, implicarà la creació d'un dataset de les dades que utilitzem.
A més, per a aquesta cerca usarem un backend FaaS conjuntament amb Lithops com a framework i interfície entre el nostre programari i el backend proporcionat per IBM.

En segon lloc, un cop obtingudes, les dades s'haurien de processar per fer-les manejables per a la nostra aplicació. Posteriorment, s'escriurien en format csv per ser emmagatzemat i així satisfer la persistència de les dades que requereix el projecte.
D'altre banda, aquest segon pas garantitzarà la consistència de les dades, i caldrà crear entitats adreçables, consistents, tolerant a errors i stateful; amb la finalitat d'estructurar les dades per facilitar la seva consulta i modificació.

En tercer lloc, mitjançant l'anàlisi de les dades que fariem a la pràctica, aprofitarem per fer una notebook amb estadístiques, gràfics de línies i de dispersió. Aquest procés, implicarà la realització de queries sobre les nostres dades, i la representació de la mateixa.

Un cop explicats els tres pilars principals del projecte, procedirem a donar-li forma a l'esquelet.
Com que el professorat ens ha donat la llibertat d'escollir la temàtica del projecte, hem volgut ser realistes amb el temps disponibles, i aprofitar aquest aprenentatge per implementar una idea que considerem prou original.

Objectius de l'aplicació

La temàtica de la nostra aplicació és clara, la seguretat informàtica. Concretament, la hem volgut enfocar a l'usuari final. La inspiració ha sorgit arran de la introducció de l'assignatura de Seguretat en Xarxes, amb el professor Jordi Castellà Roca.

Com tots sabem, la revolució d'Internet i el fet de que les nostres vides siguin parcial o completament públiques a Internet, comporta uns riscos. Concretament, en el món de la política conjuntament amb l'enginyeria social usada pels ciberatacants, pot ser útil conèixer la informació de les persones per condicionar les seves ideologies o creences. Aquest, és un fet real i que sorgeix sense que l'usuari final en sigui conscient en la majoria de casos.

Concretament, la informació amb la que treballa l'aplicació és per exemple: la localització, la geolocalització, la religió, la ideologia política, entre d'altres.
Tot plegat, s'analitza i s'usa per calcular una puntuació que li presentem a l'usuari final. Aquesta puntuació faria referència a la gravetat de la seva situació i recomanariem prendre mesures al respecte.

El missatge que li volem donar a l'usuari és que si sense dir-ho explícitament, podem predir la seva ideologia política i creença religiosa, qualsevol altre persona ho pot fer. En el nostre cas, ho fem amb bona intenció però a Internet no tothom té bones intencions; i aquesta informació pot ser usada per condicionar la seva realitat.

En aquest cas, analitzem text i els predictors treballen sobre aquest. En l'anàlisi/predicció dels diferents tweets utilitzem múltiples procesos per tal d'accelerar-ho. Però com a idea, podria implementar prediccions sobre fotografies o videos en un futur. A més, els predictors com encara tenen molt marge d'error, es pot millorar molt la seva precisió i complexitat.
Encara es poden implementar al sistema més xarxes socials i anàlisis d'altres datasets. Es poden analitzar els posts de les persones que segueix i dels seus segudiors. De fet el que fem és l'anàlisi dels seguidors d'un perfil, partint de la base de que a les persones a les que seguim son afins a nosaltres. Les apis ens limiten el volum de dades que podem utilitzar, sobre 150 usuaris en l'api de twitter ens dóna error, dient que hem excedit el volum de dades que pot donar la api. En les proves del notebook hem aconseguit treure un màxim de 150 usuaris, tot i que per defecte l'hem deixat en 30 usuaris en la següent de codi:

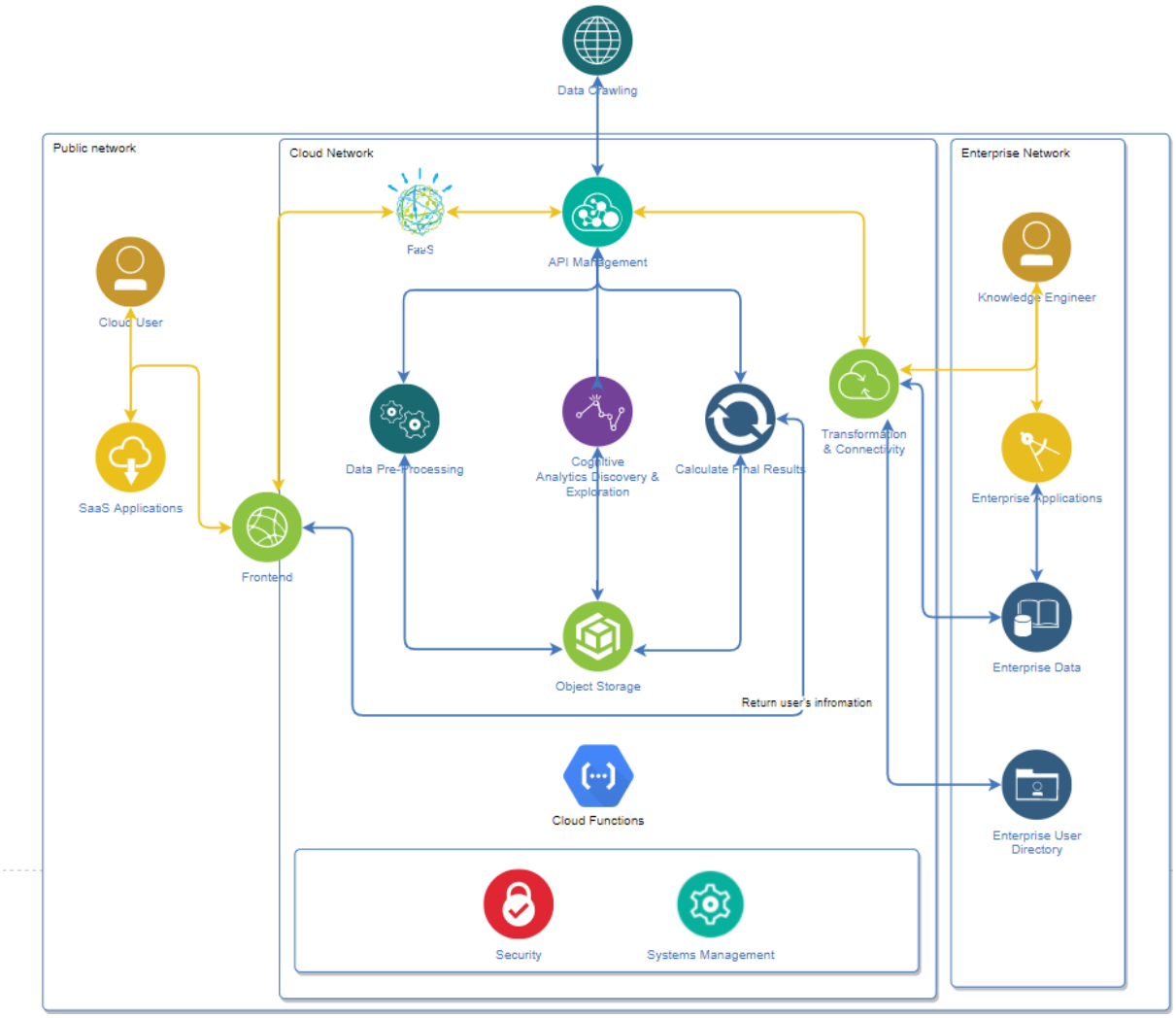
```
friends = tweepy.Cursor(api.friends, twitter_screen_name).items(30)
```

Com veiem, tractem una temàtica on la imaginació és el límit.

Per acabar, aquest apartat, cal dir que l'aplicació pel que fa a política està pensada per als Estats Units on trobem dos possibles partits polítics: demòcrates i republicans. A més, pel que fa a la religió, hem escollit 4 de les religions més importants del món.

Arquitectura

Diagrama d'arquitectura



En aquesta pràctica, tenim quatre components principals: el frontend l'aplicació en sí, el backend (IBM) i la interfície que comunica l'aplicació amb el backend (Lithops framework). Tot plegat s'encapsula en una arquitectura serverless que detallarem a continuació.

En primer lloc, el frontend no té gaire misteri ja que simplement permet que l'usuari interaccioni amb la plataforma, accedint a les seves xarxes socials mitjançant les credencials corresponents, i obtenint els resultats finals; a més d'informació complementària del projecte.

En segon lloc, l'aplicació en sí consisteix en 3 fases:

Primera fase

La primera fase consisteix en obtenir la informació de l'usuari. Un cop tenim la informació de les comptes d'aquest, procedim a fer les peticions a les dues xarxes socials que hem escollit: Twitter i Facebook. Cada una té les seves particularitats i ens permet accedir a la seva informació amb certes restriccions. Tot plegat s'ha executat mitjançant un backend FaaS d'IBM.

Segona fase

En la segona fase, un cop obtingudes les dades, les tractem per obtenir un fitxer en format csv amb tota la informació necessària. En el nostre cas, hem guardat l'identificador de l'usuari i de cada una de les seves publicacions; la seva descripció del perfil; la URL al perfil; la seva ubicació; si té activada o no la geolocalització; el contingut de cada publicació; les dates de creació, etc.
Pel que fa a la informació personal la guardem al principi del fitxer (una fila per xarxa social). Posteriorment, trobaríem totes les seves publicacions a cada línia amb la informació que hem comentat anteriorment.

A més, abans de guardar el fitxer, fem les l'anàlisi polític i religiós per guardar tota la informació en una escriptura al Object Storage d'IBM. Aquest servei ens proporciona propietats molt interessants que explicarem a l'apartat del backend.

Finalment, ens interessa retornar la informació més important a l'usuari juntament amb la puntuació que li correspon. Com hem dit anteriorment, a l'Object Storage trobarem tota la informació amb el format que desitgem. Per tant, ara serà qüestió de calcular la puntuació corresponent i reunir la informació personal que li presentarem a l'usuari.

Pel que fa a les puntuacions, hem assignat uns punts per a cada propietat que compleixi l'usuari:

Puntuació de les vulnerabilitats:

- 0-39 → Baix
- 40-69 → Mig
- 70-89 → Alt
- 90- ∞ → Crític

Les puntuacions anteriors depenen dels següents paràmetres:

- Perfil públic → +35 punts
- Geolocalització activada → +90 punts
- Localització → +5 punts
- Ideologia política != neutral → +40 punts
- Ideologia política != neutral → +40 punts

A més, a mesura que el programari comprova els paràmetres anteriors, els anirem guardant en un diccionari que li presentarem al Frontend per a que l'usuari pugui consultar les prediccions que s'han fet, i quina informació personal té pública.

D'aquesta manera, l'usuari pot tenir una referència de quina és la gravetat de la seva situació, i poder prendre mesures al respecte.

Tercera fase

En la tercera fase, hem creat un notebook que s'encarrega de fer queries sobre les dades de l'Object Storage per tal de calcular algunes estadístiques. A més, hem aprofitat la informació dels nostres predictors per representar algunes gràfiques de línies per representar tota la informació interna. Finalment, s'han usat gràfics de dispersió per donar una visió estadística encara més profunda. Tot plegat, s'ha implementat al notebook de Jupyter que trobareu al mateix repositori de la pràctica amb el nom "SocialNotebook.pdf" o "SocialNotebook.ipynb".

Aquest anàlisis estadístic s'ha fet per 3 personatges públics (Hillary Clinton, Barack Obama i Mufti Menk) amb les seves comptes de Twitter; ja que amb Facebook cal fer un Log In al compte, i per tant només podrien fer-ho els propis usuaris de la nostra aplicació. En canvi, Twitter no requereix d'aquestes credencials ja que només cal el nom d'usuari.

Backend

Propietats generals

Aquesta aplicació ha satisfet els requeriments de la pràctica que ens han proposat el professorat.

Se'ns demana dissenyar una arquitectura distribuida que garantitza la transparència a escala, la transparència de localització, la transparència d'errors, i finalment la transparència de concurrència.

- Transparència a escala: Implica que l'aplicació s'expandeix a mesura que s'incrementen les peticions, i permet fer-ho sense canviar l'estructura i els algorismes de l'app.
- Transparència de localització: Aquesta transparència implica el fet de no saber a on està el recurs al que s'accedeix.
- Transparència d'errors: Implica amagar els errors i recuperar un recurs en cas de que no estigui disponible o s'hagi corromput.
- Transparència de concurrència: Implica amagar la concurrència d'accés a un recurs remot amb altres usuaris.

Com concretarem a continuació, usarem un backend d'IBM amb FaaS i Object Storage que permet complir les propietats anteriors. El Faas i l'Object Storage ens proporcionen transparència a escala, de localització, d'errors i concurrència; ja que permeten tant a l'usuari com al programador evitar canviar l'estructura o els algorismes per escalar l'aplicació.

En cas d'actualitzar l'aplicació, l'usuari no haurà de fer-ho al seu dispositiu. A més, en cas de que a alguna de les màquines d'IBM deixi d'estar disponible, el propi cloud s'encarrega de solucionar aquesta anomalia i canviar de màquina per seguir l'execució d'aquesta.
D'altre banda, FaaS ens permet que la concurrència dels usuaris no sigui un problema, per a cada esdeveniment obrirem funcions massivament de manera paral·lela per satisfer les seves peticions.

FaaS

Per l'aplicació, executem l'entorn en un backend FaaS com hem dit.
Aquest servei és un model d'execució basat en esdeveniments i s'executa en contenidors sense estat. Aquestes funcions s'encarreguen de gestionar la lògica del programari i l'estat dels servidors. El paradigma FaaS, ens permet com a programadors dissenyar i treballar amb simplement funcions paral·leles que no han de tenir en compte el seu manteniment. En definitiva, tracta d'una abstracció per a que els desenvolupadors executem programari en resposta a certs esdeveniments.
En el nostre cas, aquests esdeveniments comencen per l'execució de l'aplicació per part de l'usuari. Per tant, si tenim un conjunt gran d'usuaris, els serveis d'IBM permetran l'execució de l'aplicació de forma massiva i paral·lela evitant centralitzar les peticions. A més l'ús d'un backend FaaS es molt apropiat pel nostre cas, ja que és un servei molt apropiat per a tasques pesades i que requereixen d'un volum de treball substancial.

Object Storage

Pel que fa a l'emmagatzematge, usem l'Object Storage d'IBM com hem dit. Per tant, hem dissenyat una plataforma amb emmagatzematge orientat a objectes que adrecem amb identificadors únics per a cada usuari. Això ho fem ja que l'usuari pot usar només una de les xarxes socials i per tant, no és adequat adreçar-lo amb amb el seu nombre d'usuari.
A més, l'Object Storage ens proporciona durabilitat, seguretat i tolerància a errors de les dades ja que ho gestiona IBM sense que el programador ni l'usuari s'hagin de preocupar del seu manteniment. Normalment, aquest servei comporta el pagament d'una quota en cas de que el volum de dades sigui substancial.

Lithops

Lithops es defineix com a framework de computació en llenguatge Python multi-cloud distribuït. Aquest, permet executar codi Python local sense modificar a gran escala en els principals Clouds com IBM, Google, AWS, entre d'altres.

Lithops pot usar-se per aplicacions com ara big data, aplicacions paral·leles, deep learning, processos de machine learning, anàlisi geoespacial entre d'altres.

Per al nostra pràctica ens ha estat molt útil. Creiem que és un framework molt potent ja que amb simplement una línia de codi ens permet escriure i llegir objectes del Cloud Object Storage; executar funcions asíncrones, entre d'altres utilitats.

Hem utilitzat dues funcions per executar les funcions dins del cloud: map i async.

La funció fexec.call_async ens permet executar una funció de forma asíncrona dins del cloud, s'executa una sola vegada i rep com parametre objectes no iterables, utilitzant el header i els paràmetres de la funció.

D'altra banda,fexec.map ens permet executar una funció sobre cada valor d'un objecte iterable.

També utilitzem la llibreria multiprocessing que ens proporciona lithops. D'aquí utilitzem pool, que ens permet paralelitzar les funcions de procés de tweets en diferents procesos.

En definitiva, ens ha permès aprofitar-nos de les propietats del cloud d'una manera senzilla i sense massa complexitat. És òptim per al que hem volgut dissenyar, i ens ha permès fer-ho d'una manera satisfactòria i eficient; disminuint responsabilitats als programadors per a que es pugui centrar en la lògica de l'aplicació.

Volum de dades

Pel que fa al volum de dades sobre el que treballem, s'ha vist incrementat substancialment un cop hem implementat la cerca dels tweets dels usuaris als que segueix l'usuari inicial. En les nostres proves del notebook hem aconseguit volums de fins a 4MB amb els perfils de la Hillary Clinton o del Barack Obama, però com hem dit anteriorment twitter imposa uns límits i per no superar els límits hem deixat un límit de cerca de 30 de seguidors.

També cal recordar que el volum sempre dependrà del nombre de posts que tingui l'usuari i el seu entorn. I en aquest cas, dels límits del framework Tweepy.

Limitacions de l'aplicació

Hem volgut dedicar unes línies a aquest apartat per aclarir alguns punts importants l'aplicació.

En primer lloc, com hem dit, predim la ideologia política i creences religioses dels usuaris. Per fer-ho, hem creat un predictor que evidentment té un marge d'errors encara gran. No es tracta de res complicat, un comptador de paraules. En aquest cas es poden usar programes més complexos per tal de minimitzar el marge d'error.

En segon lloc, degut a la naturalesa de la idea, les dades que utilitza l'aplicació per donar els resultats finals a un usuari, depèn directament de la quantitat de publicacions que tingui públiques.
A més, per a usuaris amb moltes publicacions, les APIs usades imposen un límit de dades a sol·licitar. Per evitar aquest límit, segons el nostre coneixement, cal pagar una quota en algun cas per poder accedir a un volum de dades encara més gran.
D'altre banda, pel que fa al total de dades que manipulem pot ser molt gran a l'hora de tenir multitud d'usuaris utilitzant la nostra plataforma.

Pel que fa a l'API Graph de Facebook, degut a motius legals, la seva aplicació ja no permet consultar les preferències polítiques i religioses dels seus usuaris. Pert tant, els predictors encara guanyen més importància, que no pas només per analitzar la informació de Twitter.

Conclusions:

Hem vist diverses aplicacions pràctiques de sistemes distribuïts;
Hem utilitzat diverses APIs (en el nostre cas la de facebook i twitter), hem fet un data crawler; hem utilitzat el cloud per fer les dades persistents; hem treballat sobre cloud functions; hem utilitzat lithops; i hem obtingut una visió general del funcionament del cloud.
El nostre objectiu ha estat realitzar una aplicació per fer veure als usuaris de quina manera estan exposades les seves dades a internet.

Joc de Proves

Per al joc de proves hem dissenyat un notebook com hem especificat anteriorment. Es troba al mateix repositori de la pràctica amb el nom SocialNotebook.pdf o SocialNotebook.ipynb.

Link del repositori GitHub

https://github.com/Usamasource/SD-TASK2-Big-Data