Statistiche su pubblicazioni di articoli, usando servizi AWS tramite Apache Spark

Marta Pibiri

Indice

1 Introduzione

Possiamo suddividere il progetto in quattro fasi:

- Installazione Apache Spark e librerie
- Creazione software per l'analisi di autori
- Creazione struttura cluster usando servizi AWS
- Esecuzione del progetto

In questo report verrà illustrato come installare in macchina locale il necessario per poter avviare il progetto e la creazione di una struttura cluster grazie ai servizi AWS. Il codice è scritto in Python usando librerie Pyspark, di Apache Spark. Quest'ultima è una piattaforma open source per l'elaborazione di analisi dei dati su larga scala, progettata per essere veloce e generica. Date le sue caratteristiche è possibile creare una struttura cluster composta da nodo master e da slaves per poter distribuire il calcolo. L'obbiettivo generale è stato costruire strutture dati dalle quali poter effettuare analisi sul comportamento degli autori rispetto alle loro collaborazioni nel corso degli anni, in particolare nel 2020, complice lo studio del virus COVID-19.

2 Struttura progetto

È possibile scaricare il progetto tramite interfaccia GitHub con il pulsante “Code” oppure tramite comando wget su terminale:

```
wget https://github.com/tatalessap/big_data_authors/archive/master.zip
```

Dopo aver scaricato il file zip, bisogna estrarlo, ottenendo la cartella *”big_data_authors-master”*. Al suo interno vi sono i file necessari per poter avviare il progetto, organizzati in due sottocartelle.

- *files*: files necessari per poter avviare i test,
- *terraform_bd*: files necessari per la creazione delle istanze tramite Terraform.

Successivamente è necessario aprire il seguente link a MEGA e selezionare ”Scarica ZIP”. In seguito è necessario estrarre i file scaricati all’interno della sotto cartella *files* creata in precedenza. Da terminale:

```
unzip -d big_data_authors-master/files/ Downloads/bigdata.zip
```

Questo conclude la fase di download dei files necessari per effettuare una run. Sono costruiti opportunamente a partire da un dataset iniziale che viene descritto nella prossima sezione.

3 Dataset iniziale

I dati di partenza sono composti da un file *papers_covid.json* e da una cartella *authors* dove al suo interno vi sono un insieme di file json (118.797 oggetti). Ognuno di questi file rappresenta un autore e al suo interno vi sono tutte le informazioni relative agli articoli scientifici scritti insieme ai suoi collaboratori.

- *papers_covid.json*: insieme di informazioni relativi ai papers scritti sul Sars (e sul Covid-19). Partono dagli anni 2000.
- *authors*: insieme di informazioni relativi ai singoli autori e papers scritti da essi a partire dal 2015 fino al 2020.

I file sono in tutto 118.798 e la cartella ha un peso totale di 13,5 GB. Per poter scaricare il dataset è possibile procedere da linea di comando tramite

```
wget https://aida.kmi.open.ac.uk/downloads/network_data_codiv.zip
```

4 Installazione Apache Spark e librerie - macchina locale

4.1 Installazione Spark

Per poter installare Spark da terminale, procediamo ad effettuare update e upgrade dei pacchetti di sistema:

```
sudo apt update
sudo apt -y upgrade
```

Il secondo passo è installare Java:

```
sudo apt install default-jdk
```

Ora procediamo con la fase di installazione di Apache Spark. Scarichiamo il file tgz, estraiamolo e spostiamo la cartella dove desideriamo avere la cartella Spark:

```
curl -O https://www.apache.org/dyn/closer.lua/spark/spark-2.4.5/spark-2.4.5-bin-
   hadoop2.7.tgz
tar xvf spark-2.4.5-bin-hadoop2.7.tgz
sudo mv spark-2.4.5-bin-hadoop2.7/ /opt/spark
```

Infine sistemiamo lo Spark-enviroment aprendo il file /.bashrc con nano o vim e aggiungiamo le seguenti righe:

```
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Nella prima riga inserite il percorso in cui avete messo la cartella spark. Per attivare i cambiamenti basterà utilizzare il comando:

```
source ~/.bashrc
```

4.2 Installazione libreria Graphframes

Scaricare il pacchetto da terminale:

```
wget http://dl.bintray.com/spark-packages/maven/graphframes/graphframes/0.8.1-
   spark2.4-s_2.11/graphframes-0.8.1-spark2.4-s_2.11.jar
```

Spostare il file all'interno della cartella jars di Spark:

```
mv graphframes-0.8.1-spark2.4-s_2.11.jar spark/jars
```

Per poter usare la libreria (anche in altri progetti) è consigliato aggiungerla manualmente tramite comando (all'interno del proprio codice), dove sc rappresenta il contesto Spark dichiarato in precedenza.

```
sc.addPyFile("spark/jars/graphframes-0.8.1-spark2.4-s_2.11.jar")
```

5 Utilizzo e illustrazione del modulo *authors_analysis.py*

Il modulo creato ha un insieme di metodi per eseguire le seguenti operazioni:

- creazione del file di supporto (necessaria per le parti successive)
- creazione del grafo rappresentante le collaborazioni tra autori
- creazione delle strutture relative agli autori e alle loro collaborazioni
- creazione della tabella con riportati valori e informazioni relativi al comportamento dell'autore nel corso degli anni

Nei prossimi paragrafi, verranno riportate le righe di codice per poter eseguire e creare tali strutture.

5.1 Creazione del file di supporto

Il file di supporto è un file csv (nome: *covid_plus_2015_2020.csv*) rappresentante una tabella dove vengono riportati solo gli id dei paper, DOI, l'anno e la lista degli autori partecipanti. I paper sono estratti dalla cartella authors, per questo motivo vanno dal 2015 al 2020.

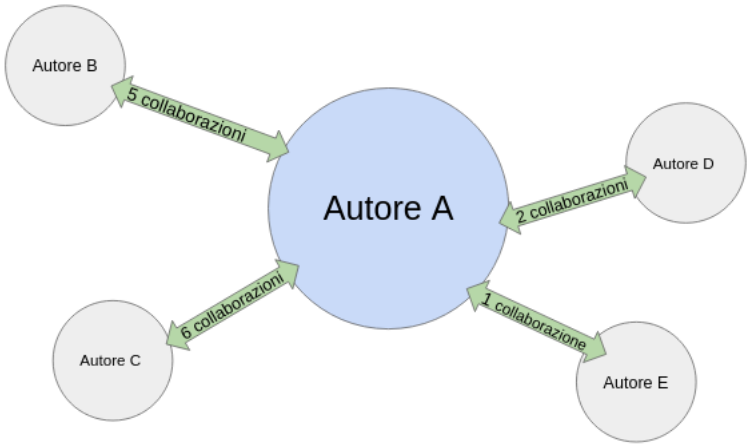
```
name=covid_plus_2015_2020
create_table_papers('network_data_codiv/authors', name=name)
```

Il file di supporto è all'interno della cartella del progetto files/csv.

5.2 Creazione del grafo

Un grafo è composto in questo modo:

- *vertici*: autori
- *archi*: se esiste un arco tra un autore A e un autore B vuol dire che esiste almeno una collaborazione tra essi. Il peso degli archi è dato dal numero di collaborazioni tra i due autori



La creazione del grafo è impostata in modo tale che venga considerato un solo anno. Per poter creare un grafo è necessario caricare come Dataframe di pyspark il file *covid_plus_2015_2020.csv*. Inoltre è possibile imporre un limite di paper presi in analisi. Esempio: se volessimo prendere i primi 30000 paper dell'anno 2019, basta impostare la variabile limit in questo modo:

```
limit = 30000 # =0 nessun limite di paper presi
year = 2019
```

Se non si desidera imporre il limite e dunque considerare tutti i paper all'interno di un anno, basta settare la variabile a zero. Per poter ottenere più grafi relativi a più anni e confrontarli, esempio anni 2020, 2019 e 2015, è possibile inserire i grafi all'interno di un dizionario python e come chiave imporre l'anno di riferimento.

```
limit = 30000
year = 2019
csv = "csv/covid_plus_2015_2020.csv"
df = spark.read.option("header", True).csv(csv)
list_g = dict()
list_g[year] = steps(df, year, limit)
```

All'interno del modulo sono state inserite le funzioni di salvataggio e caricamento dei dati.

```
# Save graph
save_graph(list_g[2019], "30000_2019")

# Load graph
list_g[2019] = load_graph("30000_2019")
```

Per poter visualizzare gli istogrammi con riportati (per ogni grafo):

- numero di vertici
- numero di archi
- degree medio
- densità del grafo $= \frac{2L}{n(n-1)}$ dove L è il numero degli archi e n il numero di vertici

è necessario richiamare la funzione:

```
visualize_graphs(list_g)
```

5.2.1 Esempio visualizzazione istogrammi

I grafi analizzati negli istogrammi sono:

- grafo 2015 con limite 30000
- grafo 2019 con limite 30000
- grafo 2020 con limite 30000

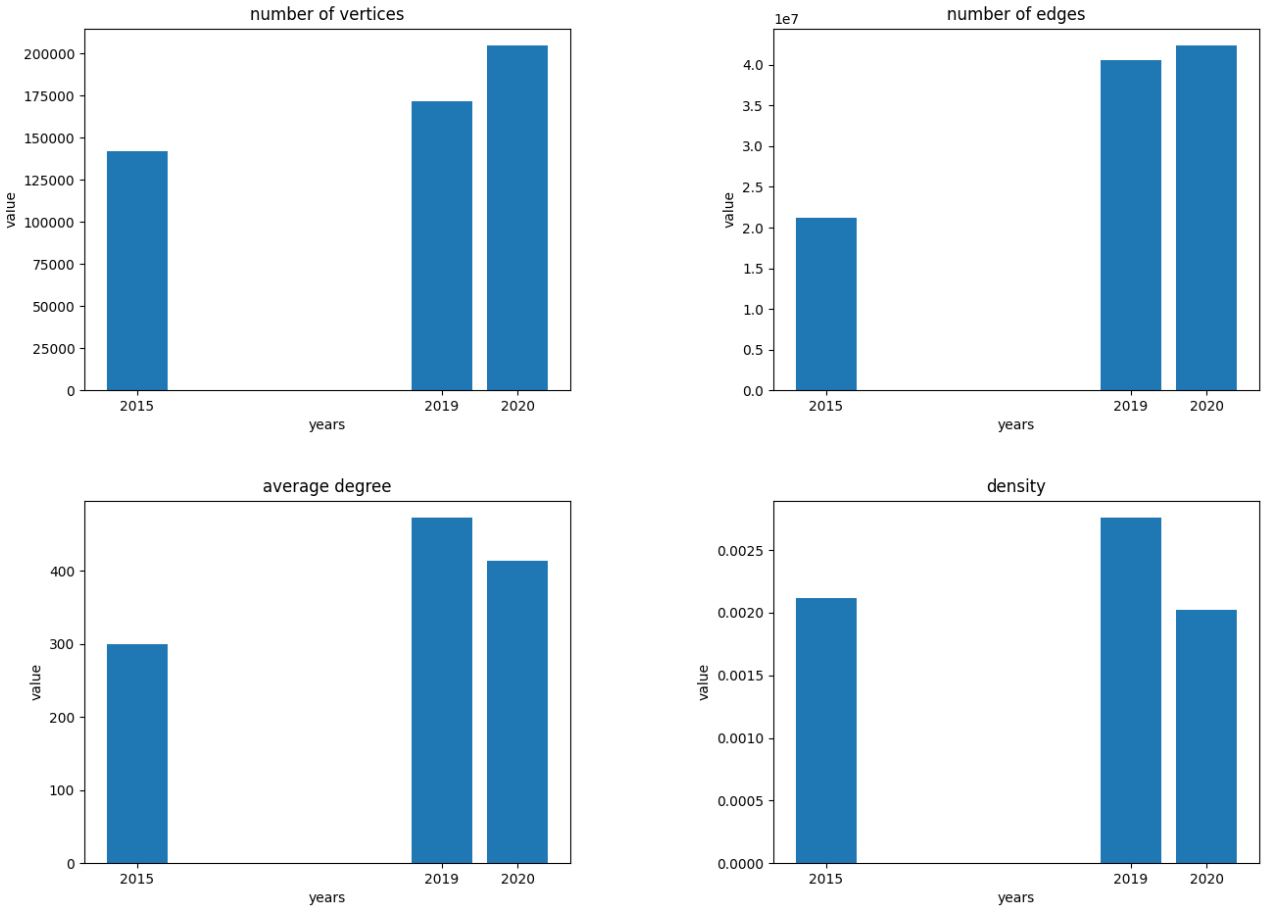


Figure 1: I quattro istogrammi

5.3 Analisi e estrazione di informazioni sugli autori

Possiamo considerare questa parte in modo indipendente rispetto alla creazione dei grafi. La creazione delle struttura relative alle collaborazioni sono fondamentali per la creazione della tabella con riportate le informazioni relative ai bienni per ogni autore.

5.3.1 Creazione *authors_collaborators*

Con *authors_collaborators* indichiamo una struttura dati a più livelli:

- primo livello: autori
- secondo livello: anno in cui ha scritto l'autore
- terzo livello: tutti i collaboratori di tutti gli anni dell'autore considerato
- quarto livello: numero di volte per cui hanno collaborato

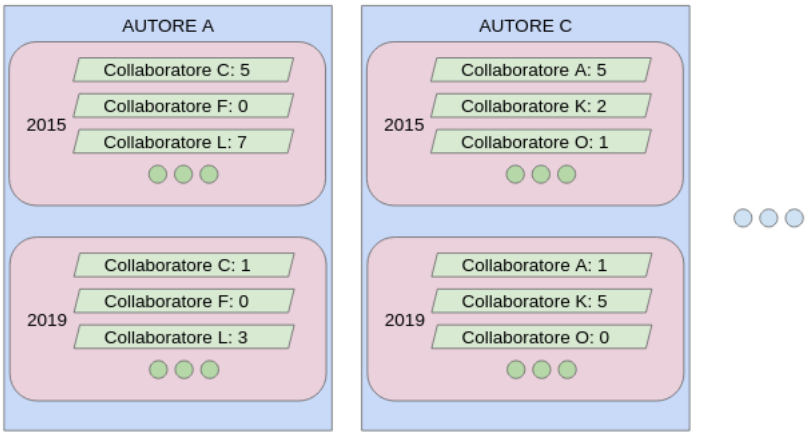


Figure 2: Struttura *authors_collaborators*

5.3.2 Creazione *authors_info*

Con *authors_info* indichiamo una struttura dati a più livelli:

- primo livello: autori
- secondo livello: nome autore e insieme di anni in cui hanno scritto paper
- terzo livello: numero di paper scritti in quell'anno

Le due strutture vengono create tramite righe di codice:

```
path = 'network_data_codiv/authors/'
list_of_json = os.listdir(path)
authors_collaborators, authors_info = collaborations_and_info_extraction({}, {},
    path, list_of_json)
```

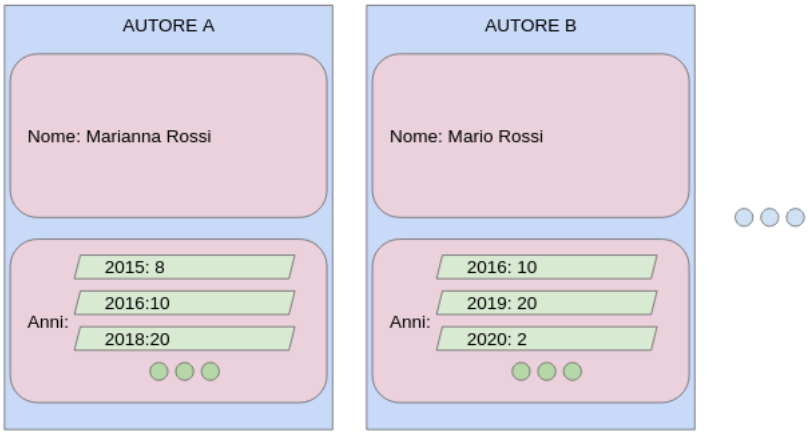


Figure 3: Struttura *authors_info*

E' possibile salvare le strutture in formato pickle per poterle caricare in seguito.

```
save_struct_authors(authors_collaborators, authors_info, 'authors_collaborators',
    'authors_info')
authors_collaborators, authors_info = load_struct_authors('authors_collaborators'
    , 'authors_info')
```

I file *authors_collaborators.pickle* e *authors_info.pickle* sono presenti nella cartella del progetto all'interno della cartella files.

5.4 Creazione della tabella con le informazioni relativi ai bienni sugli autori

La tabella sarà salvata in formato csv e le sue colonne saranno:

- id autore
- nome
- numero collaborazioni totali

Per ogni biennio (esempio 2015-2016), sono presenti le colonne:

- cosine similarity tra i due anni considerati
- numero collaborazioni nei due anni
- incremento delle collaborazioni rispetto ai due anni precedenti
- incremento delle collaborazioni rispetto a tutti gli anni
- numero paper in quei due anni

Per poter creare la tabella è necessario caricare le due strutture dati *authors_collaborators* e *authors_info* tramite la funzione *load_struct_authors*, come descritto nella sezione precedente.

```
# indicare le coppie di anni da cui si vuole creare tale analisi
list_couple_years =[(2015, 2016),
                    (2016, 2017),
                    (2017, 2018),
                    (2018, 2019),
                    (2019, 2020)]

name_csv="data_all"
create_data_authors_info(authors_collaborators, authors_info, list_couple_years,
   name_csv)
```

6 Creazione struttura cluster

Servizi AWS utilizzati:

- EC2: creazione del cluster e delle macchine in remoto
- S3 bucket: storage dei dati

Infrastrutture utilizzate:

- Terraform: creazione e impostazione delle macchine EC2

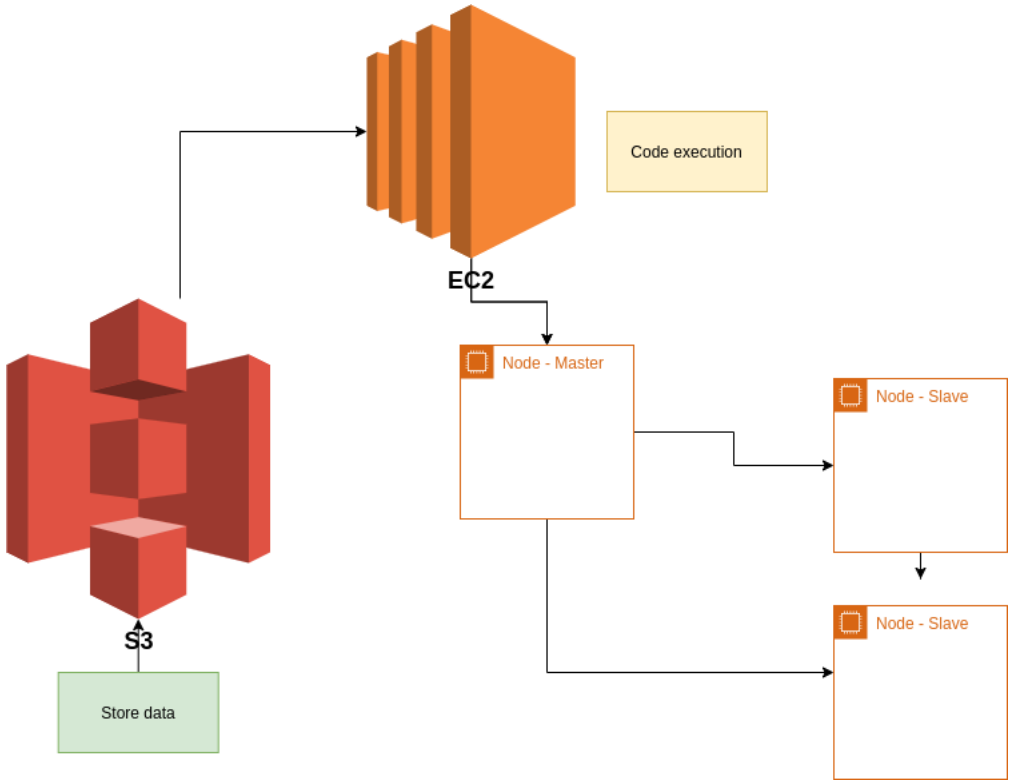
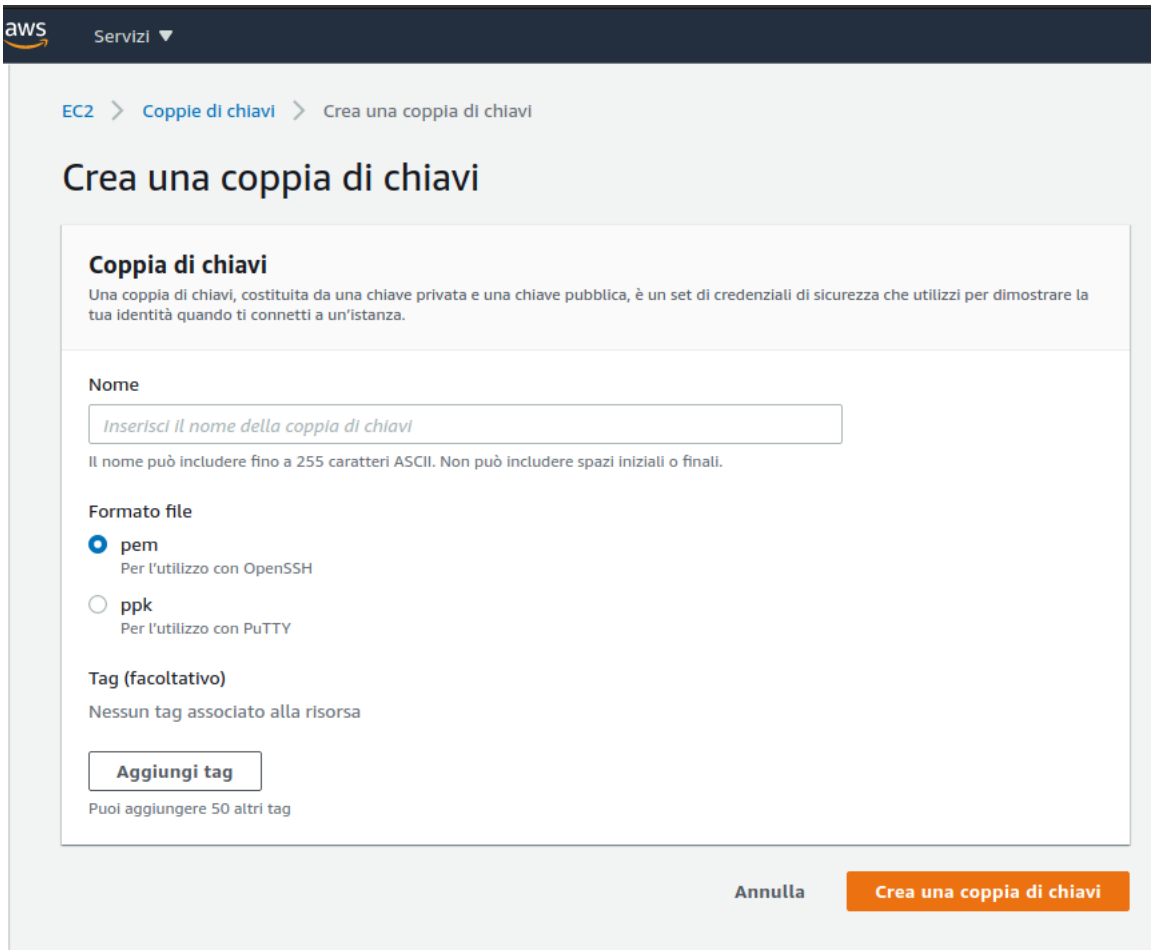


Figure 4: Schema sintetico della struttura del cluster

6.1 Creazione coppia di chiavi

Per poter utilizzare le macchine EC2 da remoto, è necessario creare una coppia di chiavi per usare il protocollo di sessione SSH. Dalla console di AWS, si deve accedere al servizio EC2 e selezionare, nella sezione Risorse, "Coppie di chiavi": in alto a destra si troverà il pulsante "Crea una coppia di chiavi". Una volta scelto un nome alla chiave (nel progetto viene chiamata my_k.pem) e selezionato come formato "pem" si può procedere a creare una coppia di chiavi e conservare il file .pem.



È necessario cambiare i permessi al file "my_k.pem" da terminale

```
chmod 400 key_bigdata.pem
```

e in seguito spostarlo nella cartella terraform_bd.

6.2 Configurazione aws-cli e credenziali - macchina locale

Si può installare aws-cli da terminale col comando

```
sudo apt install awscli -y
```

Per poterlo configurare inserire il comando:

```
aws configure
```

in modo tale da creare il file .aws/credentials. Per poterlo compilare è necessario accedere ai dettagli del proprio account AWS (Figura 6), disponibili dalla pagina di status (Figura 5).

Dalla schermata in Figura 6 si deve procedere a selezionare Show e copiare il contenuto mostrato nel file .aws/credentials aprendolo nella nostra macchina locale con il comando nano:

```
nano ~/.aws/credentials
```

Figure 5: Status Account AWS

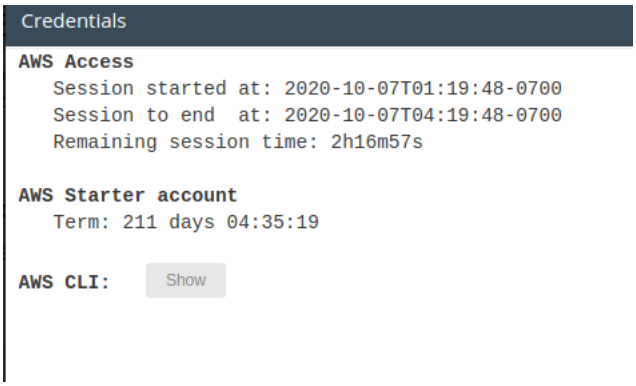


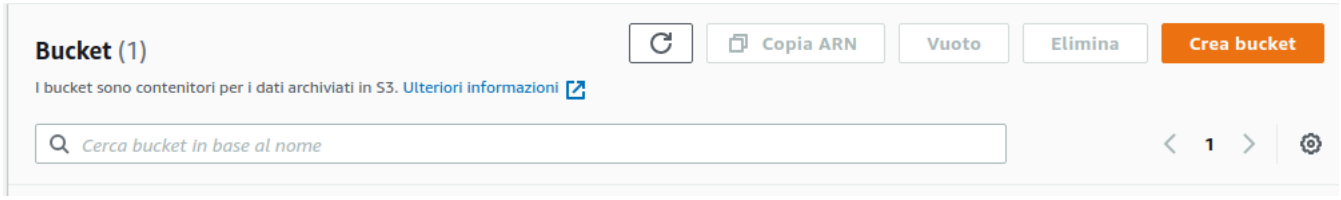
Figure 6: Dettagli Account AWS

6.3 Creazione del bucket su S3

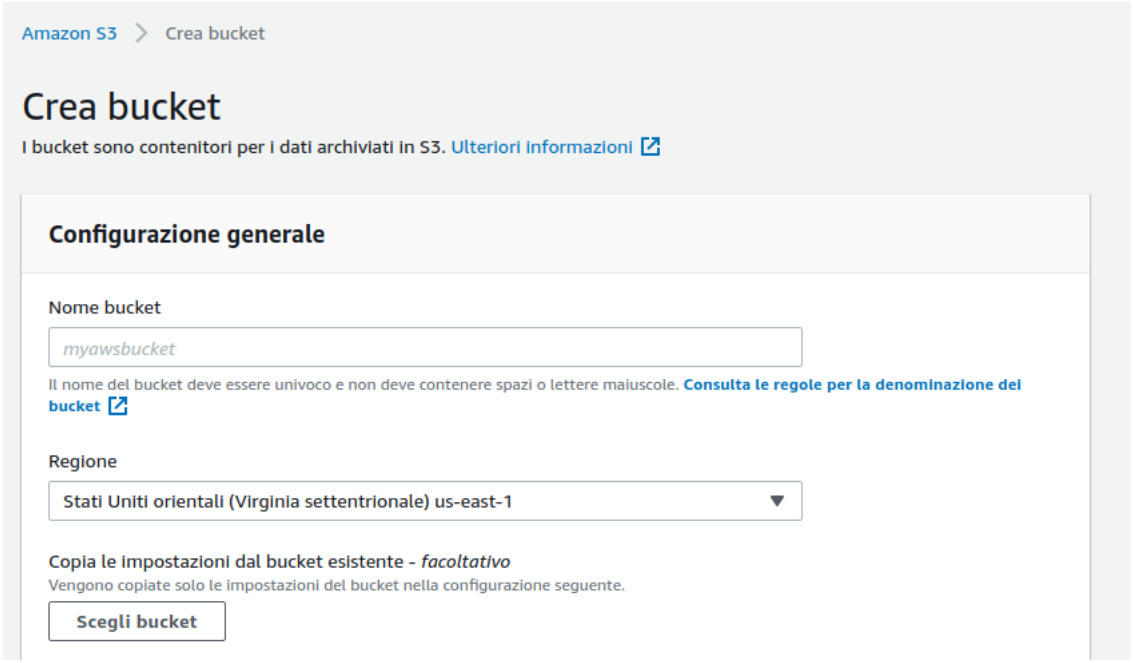
La creazione del bucket viene effettuata tramite la sezione Servizi (da cui si può accedere facilmente dopo essere entrati nella propria console AWS).

1. Cercare S3:

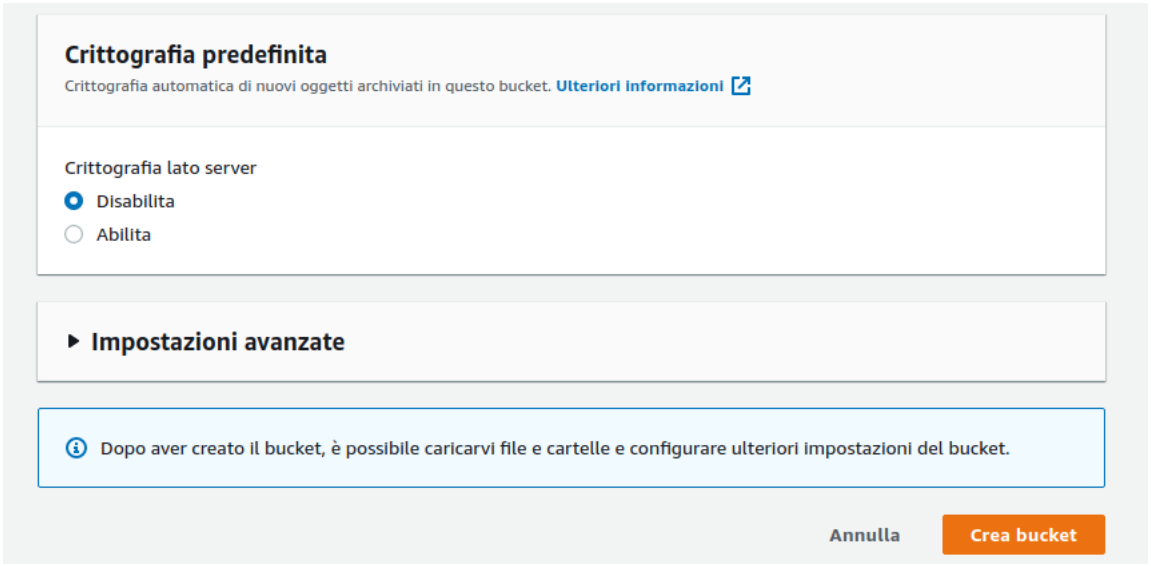
2. Selezionare Crea Bucket



3. Dare un nome al bucket (nel nostro caso chiamiamolo authorscovid):



4. Scorrere verso il basso e selezionare Crea bucket:



6.4 Caricamento file su S3

Il caricamento dei file sul bucket S3 si può effettuare tramite interfaccia oppure tramite riga di comando. Per poterlo fare da interfaccia, come visto in precedenza, si deve cercare S3 tra i servizi per poi entrare nel bucket e selezionare il pulsante carica. Alternativamente,tramite comando AWS da terminale, si può usare:

```
aws s3 cp filename.txt s3://bucket-name
```

Per terminare la configurazione del bucket authorscovid, è necessario entrare nella cartella principale del progetto () Dunque, al fine di caricare i file da linea di comando, è necessario posizionarsi all'interno della cartella principale del progetto (*"big_data_authors-master"*) e eseguire il comando

```
aws s3 cp files s3://authorscovid --recursive
```

In questo modo le istanze create tramite Terraform potranno scaricare da tale bucket i file necessari per i test.

6.5 Terraform - macchina locale

Scaricare il file zip (tramite wget) dell'installazione di Terraform e estrarlo tramite comandi:

```
wget https://releases.hashicorp.com/terraform/0.13.4/terraform_0.13.4_linux_amd64
    .zip
unzip terraform_0.13.4_linux_amd64.zip
```

Cancellare il file zip e rinominare la cartella terraform:

```
rm terraform_0.12.24_linux_amd64.zip
mv terraform Terraform/
```

Per poter eseguire in qualsiasi cartella *terraform* aggiungere alla fine della stringa PATH nel file /etc/environment il valore:

```
:/home/ubuntu/Terraform
```

6.5.1 Comandi principali Terraform

- sudo terraform init: inizializza una Terraform working directory
- sudo terraform plan: genera e mostra un piano eseguibile
- sudo terraform apply: costruisce o cambia le infrastrutture
- sudo terraform destroy: distrugge le infrastrutture

7 Esecuzione test

Prima di iniziare, entrare nella cartella terraform_bd e verificare la correttezza del percorso della chiave .pem indicato all'interno del file 'terraform.tfvars'. Per comodità tale file viene portato all'interno della cartella terraform_bd del progetto. In tal caso è possibile lasciare il percorso indicato in questo modo:

```
access_key_name = "my_k"
access_key_path = "my_k.pem"
```

Verificare se nel file 'variable.tf' ci sia il percorso corretto dove sono salvate le credenziali di aws.

```
variable "AWS_credentials_path" {
  default     = "*mettere il percorso*/.aws/credentials"
  description = "Path di riferimento per le credenziali di AWS"
}
```

Sempre all'interno della cartella terraform_bd si può inizializzare la directory di terraform:

```
sudo terraform init
```

In seguito procedere con:

```
sudo terraform apply
```

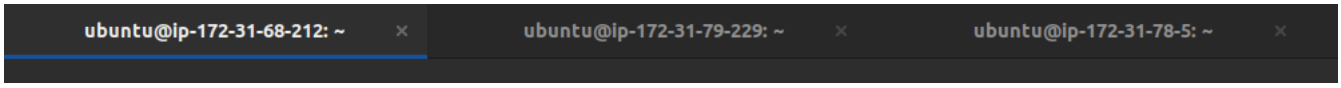
Verranno create tre macchine:

- master (con il nome namenode)
- 2 slave (con i nomi datanode2 e datanode3)

A questo punto, è possibile visualizzare tre istanze in esecuzione dal pannello di controllo EC2. Aperte tre schede nel terminale è possibile effettuare la connessione alle macchine tramite il comando:

```
ssh -i "my_k.pem" ubuntu@ec2-***.compute-1.amazonaws.com
```

Tale comando ci viene indicato anche nel pannello di controllo delle istanze su AWS, nel caso in cui le selezioniamo e premiamo il pulsante "connetti".



Dopo esserci connessi da remoto alle nostre macchine, inseriamo gli IPv4 privati delle istanze nel file /etc/hosts/:

```
sudo nano /etc/hosts
```

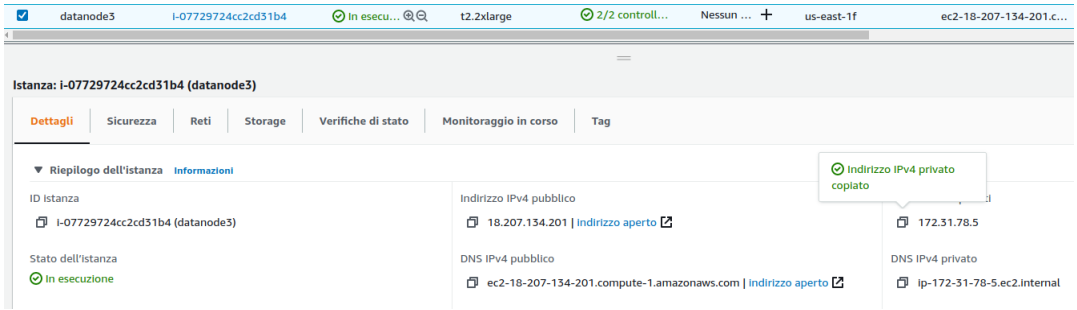
```
 GNU nano 2.9.3                                              /etc/hosts

127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

IP.MA.ST.ER namenode
IP.MA.ST.ER datanode1
IP.SL.AV.E1 datanode2
IP.SL.AV.E2 datanode3
```

Gli IPv4 li possiamo trovare nel pannello di controllo AWS delle macchine di EC2.



Dopo aver copiato gli indirizzi IPv4, il file /etc/hosts/ si presenta in questo modo:

```
127.0.0.1 localhost

# The following lines are d
::1 ip6-localhost ip6-loopb
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

172.31.68.212 namenode
172.31.68.212 datanode1
172.31.79.229 datanode2
172.31.78.5 datanode3
```

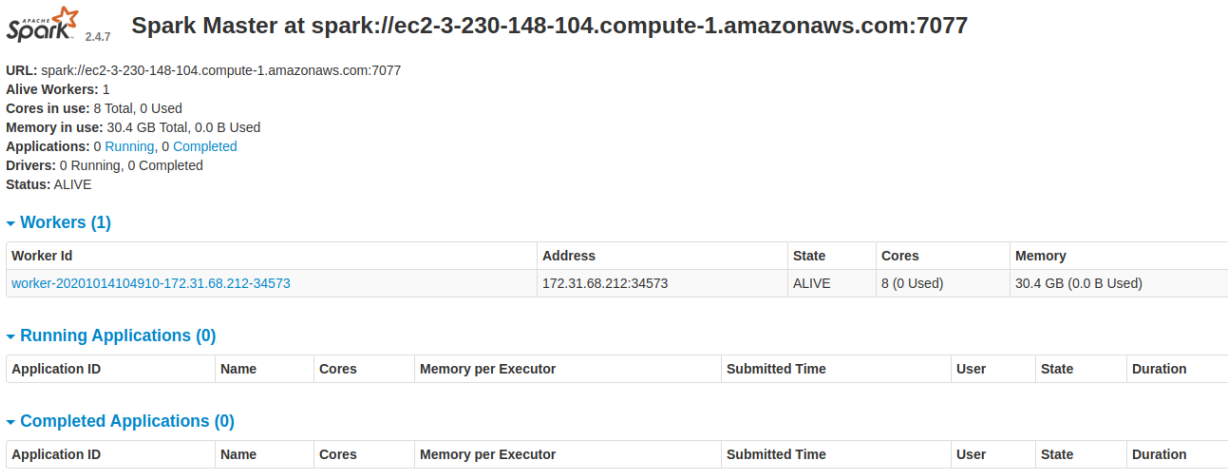
Procedere in questo modo nei file /etc/hosts/ di tutte le macchine. Dopo aver terminato questa configurazione, procedere con i comandi hdfs nel master:

```
hdfs namenode -format
$HADOOP_HOME/sbin/start-dfs.sh
$HADOOP_HOME/sbin/start-yarn.sh
$HADOOP_HOME/sbin/mr-jobhistory-daemon.sh start historyserver
```

Sempre nel master eseguire il comando di avvio di spark:

```
./spark/sbin/start-all.sh
```

Apriamo in una pagine browser nella macchina locale un indirizzo web della forma
http:/<DNSPubblicoMaster>:8080,
dove si deve sostituire <DNSPubblicoMaster> con l'effettivo DNS indicato nel pannello di controllo delle macchine EC2. La pagina web visualizzata sarà simile a questa:



Ora dalle pagine di terminale connesse alle macchine slave (datanode2 e 3) eseguiamo il comando seguente sostituendo ad URL il valore indicato nella pagina di controllo Spark.

```
./spark/sbin/start-slave.sh URL
```

Esempio:

```
./spark/sbin/start-slave.sh spark://ec2-100-25-31-217.compute-1.amazonaws.com
   :7077
```

Per far partire i test, usare il seguente comando dal terminale del master:

```
./spark/bin/spark-submit --master URL PATHFILE
```

Esempio:

```
./spark/bin/spark-submit --master spark://ec2-100-25-31-217.compute-1.amazonaws.
   com:7077 experiments.py
```

Per fermare uno slave, inserire il seguente comando nel suo terminale:

```
./spark/bin/stop-slave URL
```

Esempio:

```
./spark/bin/stop-slave spark://ec2-100-25-31-217.compute-1.amazonaws.com:7077
```

8 Risultati Test

Il test prevede la:

- creazione di un grafo con 115675 vertici e 9416896 archi.
- creazione della tabella con informazioni relative a 118797 autori

Cores	**Memory per Executor**	**Duration(minutes)**	***Workers***
48	1024.0 MB	0:04:29.951551	6
40	1024.0 MB	0:04:38.139056	5
32	1024.0 MB	0:04:33.872476	4
24	1024.0 MB	0:04:35.27364	3
16	1024.0 MB	0:04:32.331784	2
8	1024.0 MB	0:04:33.443617	1

Table 1: Risultati creazione grafo

Cores	**Memory per Executor**	**Duration(minutes)**	***Workers***
48	1024.0 MB	0:03:02.736669	6
40	1024.0 MB	0:03:03.505846	5
32	1024.0 MB	0:03:02.727538	4
24	1024.0 MB	0:03:02.497900	3
16	1024.0 MB	0:03:02.893771	2
8	1024.0 MB	0:03:05.427424	1

Table 2: Risultati creazione tabella

I risultati ottenuti sia nella creazione del grafo sia nella creazione della tabella non evidenziano un miglioramento netto all'aumentare dei worker. L'esperimento ha reso necessario l'utilizzo di determinate strutture dati hanno inciso nella non-parallelizzazione ottimale del codice, nonostante l'utilizzo di più librerie di Pyspark.

Referenze

[1] Stefano Raimondo Usai. *Creazione Istanze su AWS*.

[2] *Terraform*. https://www.terraform.io/.

[3] *AWS*. https://aws.amazon.com/it/s3/.

[4] *comandi AWS S3*. https://docs.aws.amazon.com/it_it/cli/latest/userguide/cli-services-s3-commands.html.

[5] *GraphFrames*. https://graphframes.github.io/graphframes/docs/_site/api/python/index.html.