AWSQ report

G30

November 2023

Since we are interested just in the execution time of our query in the EC2 instance, we adapted our code to print only that. You can find the code contained in "AWSQ.py" both in the main.ipynb and at the end of this report.

1 EC2 instance configuration

- Access AWS services through one of our laboratories modules.
- Search EC2 service and start an instance:"ADM_EC2".
- Select the Amazon Linux AMI.
- Create a new key: "ADM_EC2_key.pem".
- Select t.2 medium type of instance (since we need at least 1.2 GB of memory)
- Create instance and connect.
- Create a directory "ADM_Q_folder" with:

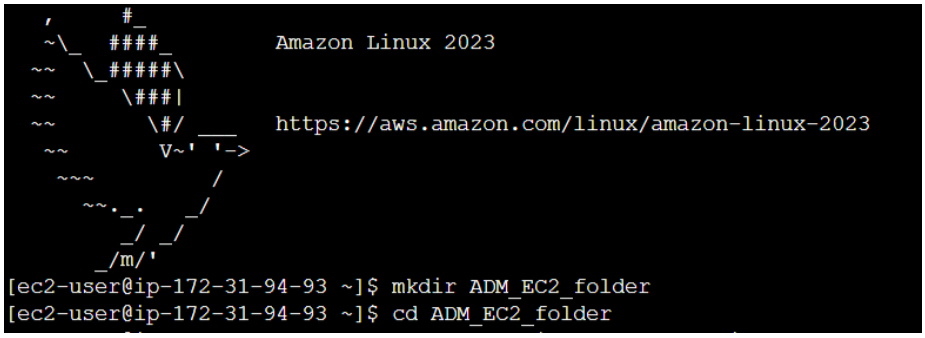

Figure 1: Create and enter folder

2 Upload data and script to the instance

We worked with FileZilla, a client providing an easy-to-use interface for transferring files between your local machine and a remote server:

- Open FileZilla → file → create a new connection, and call it "ADM_EC2_connection".
- Select SFTP as protocol (SSH File Transfer Protocol)
- Paste the ADM_EC2 public address in the "host" field.
- Select 22 as the port number.
- Select "file key" as type of access and select the ADM_EC2_key.pem file.
- Write "ec2-user" in user field.
- Connect.
- Drop "AWSQ.py" and "list.json" into the previously created ADM_EC2_folder.

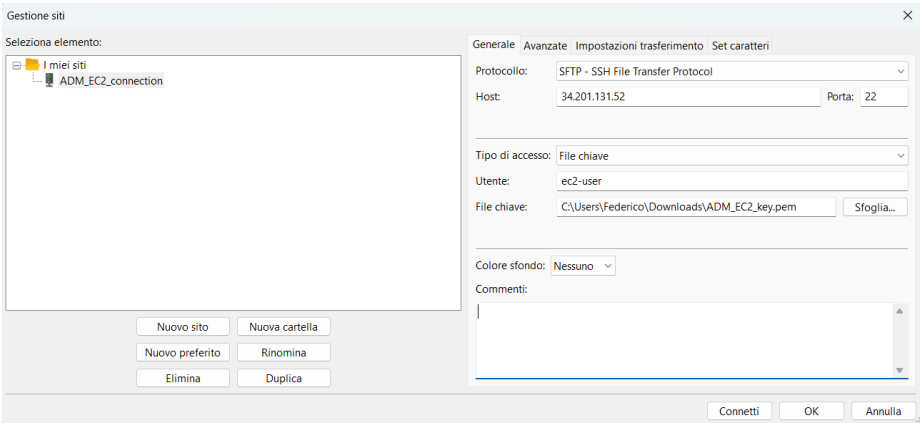

Figure 2: FileZille connection

3 Run the query

Once checked we have all the files, let's install the right libraries and finally run our query:

- Check we got all the files:

```
[ec2-user@ip-172-31-94-93 ADM_EC2_folder]$ ls
AWSQ.py  list.json
```

Figure 3: Check files

- Install pip and the needed libraries:

```
sudo yum install -y python3-pip
sudo pip3 install pandas
sudo pip3 install json5
```

- Finally run our code:

```
[ec2-user@ip-172-31-94-93 ADM_EC2_folder]$ python3 AWSQ.py
Execution time: 18.54792833328247
```

Figure 4: Query execution time

4 Conclusions

The same query "AWSQ.py" took on our local machines around 20s, as you can check on the main.ipynb notebook. Given the setup effort gaining around 2s isn't really satisfying. To really benefit from Cloud Computing we will probably need larger datasets, or harder tasks.

AWSQ.py code

```
import json
import pandas as pd
import time


initial_time = time.time()
tags_count = {}
with open("list.json", 'r') as file:
    for line in file:
        line = line.strip()
        if line:
            # Read i-th dict
            ith_dict = json.loads(line)
            if "tags" in ith_dict.keys():
                ith_tags = ith_dict["tags"]
                for tag in ith_tags:

                    # If we've already seen a certain tag update its count
                    if tag in tags_count.keys():
                        tags_count[tag] += 1

                    # If we haven't, set its count to 1
                    else:
                        tags_count[tag] = 1

# Convert it into a dataframe, sort it wrt "count", and take its first 5
    rows.
tags_count = pd.DataFrame(list(tags_count.items()), columns = ["Tag",
    "Count"])
tags_count = tags_count.sort_values("Count", ascending = False).iloc[0:5]

print(f"Execution time: {time.time() - initial_time}")
```