Homework 3

Giuliano di Giuseppe, Raffaele Russo, Alessandro Vanacore

Indice

1 Traccia

Tramite Apache Kafka gestire l'acquisizione di uno stream di dati (e.g., messaggi da un web server, misure da una rete di sensori, etc.) da una data sorgente (anche simulata) e la successiva memorizzazione in un file di log su HDFS. Utilizzando poi uno degli strumenti del primo homework (pyspark, pig o hive) effettuare delle query sullo stream di dati mostrandone i risultati su un'apposita dashboard.

2 Generazione dei dati

Per la generazione dei dati si utilizza la libreria python *psutils*, con la quale si estraggono statistiche in tempo reale dalla cpu, memoria e disco.

3 Setup Kafka

Il primo passo è avviare il server zookeeper e il server kafka con i comandi:

```
$ bin/zookeeper-server-start.sh config/zookeeper.properties
$ bin/kafka-server-start.sh config/server.properties
```

3.1 Producer

Si procede alla creazione di un topic e di un producer che può inviare messaggi su questo topic. A tal proposito si utilizza lo script python precedente per automatizzare le operazioni. Il generico messaggio contiene informazioni sul nome del topic, timestamp e statistiche sulla macchina.

Per la creazione del topic e dell'invio da parte del producer su tale topic si utilizza il codice rispettivamente in figura 1 e 2.

```python
def connect(name_topic):
    cmds = ["cd /opt/kafka_2.11-0.9.0.0"
        ,"/opt/kafka_2.11-0.9.0.0/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic "+name_topic
    ]

    try:
        for cmd in cmds:
            result = subprocess.Popen(cmd, shell=True)
            print('Comando eseguito correttamente.')
            print('Err:\n', result.stderr)

    except subprocess.CalledProcessError as ex:
        print('Esecuzione fallita.')
        print('Err:', ex.stderr)
```

Figura 1: Creazione topic

3.2 Consumer

Si utilizza un consumer che legge dal topic precedente e salva tale informazioni su un file di log di Hadoop. A tal proposito, dopo aver configurato correttamente Hadoop, si utilizza lo script bash in figura 3. Quest'ultimo avvia un producer che legge dal topic precedente e salva tale contenuto sul file di log specificato.

```python
while(1):
    cmd_prod = "/opt/kafka_2.11-0.9.0.0/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic "+name_topic
    cmd_full = "echo "+ '\"'+name_topic+" "+get_stats()+'\" | ' + cmd_prod
    cmd_mess = subprocess.run(cmd_full, shell=True, capture_output=True, text=True)
    print(f"Message {i} sent")
    sleep(0.1)
    i+=1
    if i==5000:
        break
```

Figura 2: Invio producer su topic specificato

```bash
#!/bin/bash

name_topic="launchv5"
hdfs_log_file="/home/parallels/hadoop/logs/hadoop-parallels-namenode-ubuntu-linux-22-04-desktop.log"
command="/opt/kafka_2.11-0.9.0.0/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic $name_topic --from-beginning >> $hdfs_log_file"

eval $command
```

Figura 3: Script bash consumer

3.2.1 Setup Hadoop

Per la configurazione di Hadoop si usano i comandi in figura 4. Il primo comando esegue la formattazione del NameNode di Hadoop, mentre il secondo comando avvia l'HDFS.

```python
subprocess.run("/home/parallels/hadoop/bin/hdfs namenode -format", shell = True)
subprocess.run("/home/parallels/hadoop/sbin/start-dfs.sh", shell = True)
```

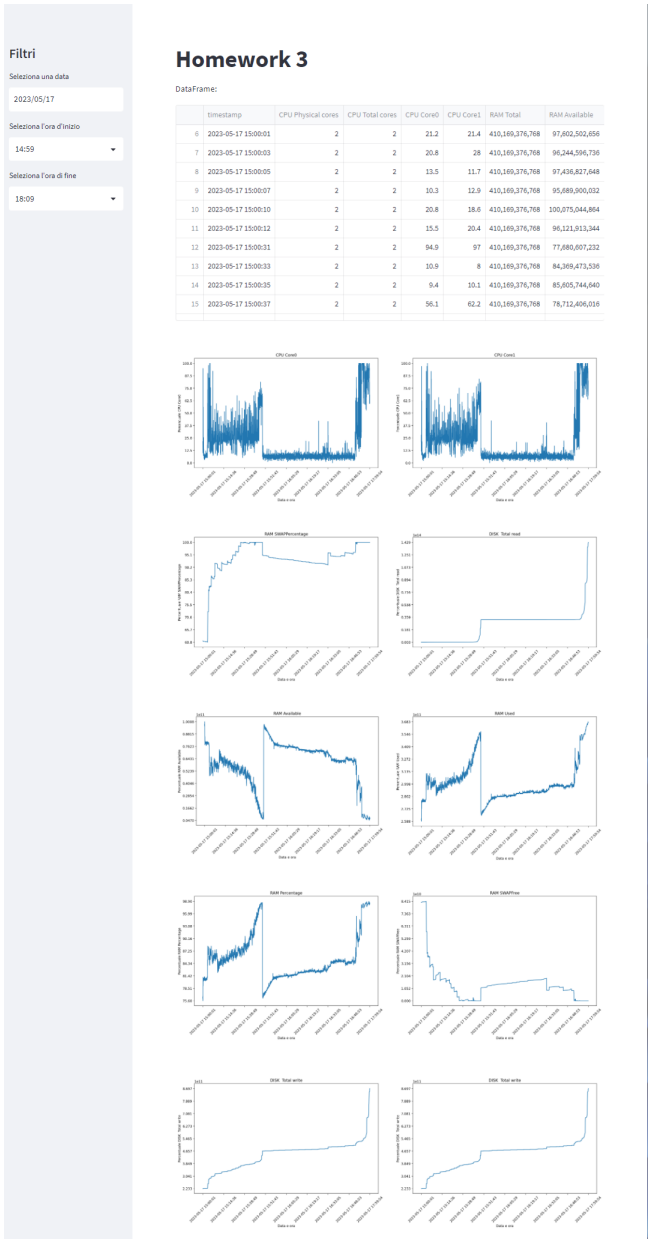
Figura 4: Hadoop

In figura 5 sono mostrate cinque righe salvate dal consumer nel file di log di Hadoop contententi rispettivamente nome del topic, timestamp e statistiche della macchina.

```
parallels@ubuntu-linux-22-04-desktop:~/hadoop/logs$ cat hadoop-parallels-namenode-ubuntu-linux-22-04-desktop.log | tail -5
launchv5 {'timestamp': '2023-05-17 16:32:35', 'CPU Physical cores': 2, 'CPU Total cores': 2, 'CPU Core0': 6.1, 'CPU Core1': 8.1, 'RAM Total': '3.82GB'
, 'RAM Available': '672.25MB', 'RAM Used': '2.72GB', 'RAM Percentage': 82.8, 'RAM SWAPTotal': '2.00GB', 'RAM SWAPFree': '177.89MB', 'RAM SWAPUsed': '1
.83GB', 'RAM SWAPPercentage': 91.3, 'DISK  Total read': '301.76GB', 'DISK  Total write': '4.48GB'}
launchv5 {'timestamp': '2023-05-17 16:32:37', 'CPU Physical cores': 2, 'CPU Total cores': 2, 'CPU Core0': 5.1, 'CPU Core1': 4.1, 'RAM Total': '3.82GB'
, 'RAM Available': '670.95MB', 'RAM Used': '2.72GB', 'RAM Percentage': 82.8, 'RAM SWAPTotal': '2.00GB', 'RAM SWAPFree': '177.89MB', 'RAM SWAPUsed': '1
.83GB', 'RAM SWAPPercentage': 91.3, 'DISK  Total read': '301.76GB', 'DISK  Total write': '4.48GB'}
launchv5 {'timestamp': '2023-05-17 16:32:39', 'CPU Physical cores': 2, 'CPU Total cores': 2, 'CPU Core0': 6.1, 'CPU Core1': 5.1, 'RAM Total': '3.82GB'
, 'RAM Available': '673.60MB', 'RAM Used': '2.72GB', 'RAM Percentage': 82.8, 'RAM SWAPTotal': '2.00GB', 'RAM SWAPFree': '177.89MB', 'RAM SWAPUsed': '1
.83GB', 'RAM SWAPPercentage': 91.3, 'DISK  Total read': '301.76GB', 'DISK  Total write': '4.48GB'}
launchv5 {'timestamp': '2023-05-17 16:32:41', 'CPU Physical cores': 2, 'CPU Total cores': 2, 'CPU Core0': 10.2, 'CPU Core1': 8.2, 'RAM Total': '3.82GB
', 'RAM Available': '675.23MB', 'RAM Used': '2.71GB', 'RAM Percentage': 82.7, 'RAM SWAPTotal': '2.00GB', 'RAM SWAPFree': '177.91MB', 'RAM SWAPUsed': '
1.83GB', 'RAM SWAPPercentage': 91.3, 'DISK  Total read': '301.76GB', 'DISK  Total write': '4.48GB'}
launchv5 {'timestamp': '2023-05-17 16:32:43', 'CPU Physical cores': 2, 'CPU Total cores': 2, 'CPU Core0': 6.1, 'CPU Core1': 5.1, 'RAM Total': '3.82GB'
, 'RAM Available': '673.16MB', 'RAM Used': '2.72GB', 'RAM Percentage': 82.8, 'RAM SWAPTotal': '2.00GB', 'RAM SWAPFree': '177.91MB', 'RAM SWAPUsed': '1
.83GB', 'RAM SWAPPercentage': 91.3, 'DISK  Total read': '301.76GB', 'DISK  Total write': '4.48GB'}
```

Figura 5: 5 righe del file di log

4 Dashboard



	timestamp	CPU Physical cores	CPU Total cores	CPU Core0	CPU Core1	RAM Total	RAM Available
6	2023-05-17 15:00:01	2	2	21.2	21.4	410,169,376,768	97,602,502,656
7	2023-05-17 15:00:03	2	2	20.8	28	410,169,376,768	96,244,596,736
8	2023-05-17 15:00:05	2	2	13.5	11.7	410,169,376,768	97,436,827,648
9	2023-05-17 15:00:07	2	2	10.3	12.9	410,169,376,768	95,689,900,032
10	2023-05-17 15:00:10	2	2	20.8	18.6	410,169,376,768	100,075,044,864
11	2023-05-17 15:00:12	2	2	15.5	20.4	410,169,376,768	96,121,913,344
12	2023-05-17 15:00:31	2	2	94.9	97	410,169,376,768	77,680,607,232
13	2023-05-17 15:00:33	2	2	10.9	8	410,169,376,768	84,369,473,536
14	2023-05-17 15:00:35	2	2	9.4	10.1	410,169,376,768	85,605,744,640
15	2023-05-17 15:00:37	2	2	56.1	62.2	410,169,376,768	78,712,406,016

Figura 6: Dashboard