



3.25 Proyecto final.  
28 de Noviembre de 2024.

## TRABAJO

Materia: Cómputo paralelo y distribuido.

Equipo:

Axel Ricardo Moncloa Muro

Leonardo Trevizo Herrera

- Utilizar al menos dos API's (Airlabs, Nasa, Openweather, etc.)

```

SITE B.sql  SITE A.sql  main.py  group.py  docker-compose.yml  SIT
proyecto final > main.py > ...
1  from kafka import KafkaProducer, KafkaConsumer
2  import json, requests, time, threading, logging
3  from collections import defaultdict, Counter
4  from pymongo import MongoClient
5  import matplotlib.pyplot as plt
6  import seaborn as sns
7
8  inicio = time.time()
9
10 POKEMON_URL = "https://pokeapi.co/api/v2/characteristic/"
11 USER_URL = "https://randomuser.me/api"
12 contador = 1
13

```

- Crear en Python un productor de mensajes de las API's

```

10 POKEMON_URL = "https://pokeapi.co/api/v2/characteristic/"
11 USER_URL = "https://randomuser.me/api"
12 contador = 1
13
14 producer = KafkaProducer(
15     bootstrap_servers='localhost:9092',
16     value_serializer=lambda v: json.dumps(v).encode('utf-8')
17 )
18
19 client = MongoClient("mongodb://localhost:27017/")

```

- Mediante Apache Kafka administrar los mensajes (crear un topic por cada API-al menos dos-)

```
C:\Users\Leonardo T H>docker pull apache/kafka
Using default tag: latest
latest: Pulling from apache/kafka
25f62819a77f: Download complete
0efe887b0486: Download complete
43c4264eed91: Download complete
2a485d968122: Download complete
4fb0eeeb44ea: Download complete
bf937160a439: Download complete
6be5fec448e3: Download complete
fef81e99f8f4: Download complete
c4d7b6d867a4: Download complete
60756e0c14ce: Download complete
Digest: sha256:22c4bea38875408e8f9fe52aca8e3a6ee67f9aa0090db59af99a2f6647558db5
Status: Downloaded newer image for apache/kafka:latest
docker.io/apache/kafka:latest

C:\Users\Leonardo T H>docker pull confluentinc/cp-zookeeper
Using default tag: latest
latest: Pulling from confluentinc/cp-zookeeper
dd9706dec42d: Download complete
ccc2996f86eb: Download complete
5c8393feadbd: Download complete
4502da6dae22: Download complete
1426f236e762: Download complete
1829166f77fd: Download complete
095c1a026149: Download complete
6a1f3bc458ce: Download complete
083d2de0f6e6: Download complete
a6437f631a5c: Download complete
daaaba019b7d: Download complete
45d49d5a2b20: Download complete
bd20a306a5d4: Download complete
Digest: sha256:01cebe9b03be37e5ccc3fee01c54ca8b0a104eda4f2380c031aa5c34ba799f8
Status: Downloaded newer image for confluentinc/cp-zookeeper:latest
docker.io/confluentinc/cp-zookeeper:latest

C:\Users\Leonardo T H>cd C:\Users\Leonardo T H\OneDrive\Documentos\ESCUELA\UACH\9no SEMESTRE\Computo Paralelo y Distribuido\parcial3\proyecto final
C:\Users\Leonardo T H\OneDrive\Documentos\ESCUELA\UACH\9no SEMESTRE\Computo Paralelo y Distribuido\parcial3\proyecto final>docker exec -it kafka bash
[+] Running 3/3
  █ kafka Pulled                                16.7s
  █ 7e24c234f4e4 Download complete              0.4s
  █ 6af9c43304ac Download complete              4.4s
  █ Network proyectofinal_default Created       0.0s
  █ Container zookeeper Started                0.3s
  █ Container kafka Started                    0.3s

C:\Users\Leonardo T H\OneDrive\Documentos\ESCUELA\UACH\9no SEMESTRE\Computo Paralelo y Distribuido\parcial3\proyecto final>docker exec -it kafka bash
[appuser@61398c467924 ~]$ kafka-topics --create --topic topico-a --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
Created topic topico-a.
[appuser@61398c467924 ~]$ kafka-topics --create --topic topico-b --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
Created topic topico-b.
[appuser@61398c467924 ~]$ kafka-console-consumer --topic-a --from-beginning --bootstrap-server localhost:9092
```

```

50 def obtener_data_api_pokemon(url:str, params:None):
51     print(POKEMON_URL+str(params)+"/")
52     response = requests.get(POKEMON_URL+str(params))
53     if response.status_code == 200:
54         return response.json() # Devuelve datos en formato JSON
55     else:
56         print(f"Error: {response.status_code}")
57         return None
58
59 def obtener_data_api_user(url:str):
60     response = requests.get(USER_URL)
61     if response.status_code == 200:
62         return response.json().get('results')[0] # Devuelve datos en formato JSON
63     else:
64         print(f"Error: {response.status_code}")
65         return None
66
67 while True:
68     actual = time.time()
69     pokemon = obtener_data_api_pokemon(POKEMON_URL,contador)
70     #user = requests.get(user_url)
71     if pokemon:
72         productor.send('topico-a', value=pokemon) # Envía datos al topic 'topic-a'
73         #print(f"Enviado a topico-a:\n {pokemon}")
74         print(f"Enviado a topico-a:\n")
75         time.sleep(3.5)
76     user = obtener_data_api_user(USER_URL)
77     if user:
78         productor.send('topico-b', value=user)
79         #print(f"Enviado a topico-b:\n {user}")
80         print(f"Enviado a topico-b:\n")
81         time.sleep(5.5)
82     pokemon=None
83     user=None
84     contador +=1
85     if (actual-inicio) > 40:
86         break

```

```

SITE B.sql  SITE A.sql  main.py  group.py  docker-compose.yml  SITE - REPLICA.sql
proyecto final > docker-compose.yml
1  version: '3.8'
2  services:
3      zookeeper:
4          image: confluentinc/cp-zookeeper:latest
5          container_name: zookeeper
6          ports:
7              - "2181:2181"
8          environment:
9              ZOOKEEPER_CLIENT_PORT: 2181
10             ZOOKEEPER_TICK_TIME: 2000
11
12     kafka:
13         image: confluentinc/cp-kafka:latest
14         container_name: kafka
15         ports:
16             - "9092:9092"
17         environment:
18             KAFKA_BROKER_ID: 1
19             KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
20             KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
21             KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092
22             KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
23         depends_on:
24             - zookeeper
25

```

- Crear en Python un consumidor que tome los datos del topic de Apache Kafka

```

23
24 consumer = KafkaConsumer(
25     'topico-a', 'topico-b',
26     bootstrap_servers=['localhost:9092'],
27     auto_offset_reset='earliest',
28     enable_auto_commit=True,
29     value_deserializer=lambda x: json.loads(x.decode('utf-8'))
30 )
31
32 consumerA = KafkaConsumer(
33     'topico-a',
34     bootstrap_servers=['localhost:9092'],
35     auto_offset_reset='earliest',
36     enable_auto_commit=True,
37     value_deserializer=lambda x: json.loads(x.decode('utf-8'))
38 )
39
40 consumerB = KafkaConsumer(
41     'topico-b',
42     bootstrap_servers=['localhost:9092'],
43     auto_offset_reset='earliest',
44     enable_auto_commit=True,
45     value_deserializer=lambda x: json.loads(x.decode('utf-8'))
46 )
47
48

```

- Almacenar los datos en MongoDB

```

88
89 counter=0
90 for mensaje in consumerA:
91     #print(f"Insertando en MongoDB: {datos}")
92     if counter<=20:
93         datos = mensaje.value # Datos recibidos del topic
94         print(f"Insertando en MongoDB A")
95         coleccionA.insert_one(datos) # Inserta los datos como un documento en la colección
96     else:
97         break
98     counter+=1
99
100
101 counter=0
102 for mensaje in consumerB:
103     if counter<=20:
104         datos = mensaje.value # Datos recibidos del topic
105         #print(f"Insertando en MongoDB: {datos}")
106         print(f"Insertando en MongoDB B")
107         coleccionB.insert_one(datos) # Inserta los datos como un documento en la colección
108     else:
109         break
110     counter+=1
111
112 datos_pokemon = list(coleccionA.find())
113 datos_usuario = list(coleccionB.find())
114

```



- Tomar los datos de MongoDB y graficar los resultados

```

111
112 datos_pokemon = list(coleccionA.find())
113 datos_usuario = list(coleccionB.find())
114
115 valores_pokemon = [dato['highest_stat']['name'] for dato in datos_pokemon if 'highest_stat' in dato]
116 #print(valores_pokemon)
117
118 valores_usuarios = [dato['results'][0]['nat'] for dato in datos_usuario if 'results' in dato]
119 #print(valores_usuarios)
120
121 # Contar las nacionalidades
122 nationality_counts = Counter(valores_usuarios)
123 nationality_labels = list(nationality_counts.keys())
124 nationality_values = list(nationality_counts.values())
125
126 # Contar valores de highest_stat
127 stat_counts = Counter(valores_pokemon)
128 stat_labels = list(stat_counts.keys())
129 stat_values = list(stat_counts.values())
130
131 # Crear la figura con subplots
132 fig, axes = plt.subplots(1, 2, figsize=(14, 6)) # 1 fila, 2 columnas
133
134 # Gráfica de nacionalidades
135 axes[0].bar(nationality_labels, nationality_values, color='skyblue')
136 axes[0].set_title("Distribución de Nacionalidades", fontsize=16)
137 axes[0].set_xlabel("Nacionalidad", fontsize=12)
138 axes[0].set_ylabel("Frecuencia", fontsize=12)
139 axes[0].grid(axis='y', linestyle='--', alpha=0.7)
140
141 # Gráfica de highest_stat
142 axes[1].bar(stat_labels, stat_values, color='orange')
143 axes[1].set_title("Distribución de Highest Stat", fontsize=16)
144 axes[1].set_xlabel("Highest Stat", fontsize=12)
145 axes[1].set_ylabel("Frecuencia", fontsize=12)
146 axes[1].grid(axis='y', linestyle='--', alpha=0.7)
147
148 # Ajustar el diseño
149 plt.tight_layout()
150
151 # Mostrar ambas gráficas
152 plt.show()
153

```

## Resultados

