CLOUD-COMPUTING
APACHE KAFKA + SPARK STREAMING + PYSPARK

Student: Shih-Ching, Huang

ID: 19631

Professor: Henry Chang

TA: Gu Liang

Course: CS570 - Big Data Processing & Analytics

TABLE OF CONTENT

INTRODUCTION

Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation, written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs but also provides the PySpark shell for interactively analyzing your data in a distributed environment.

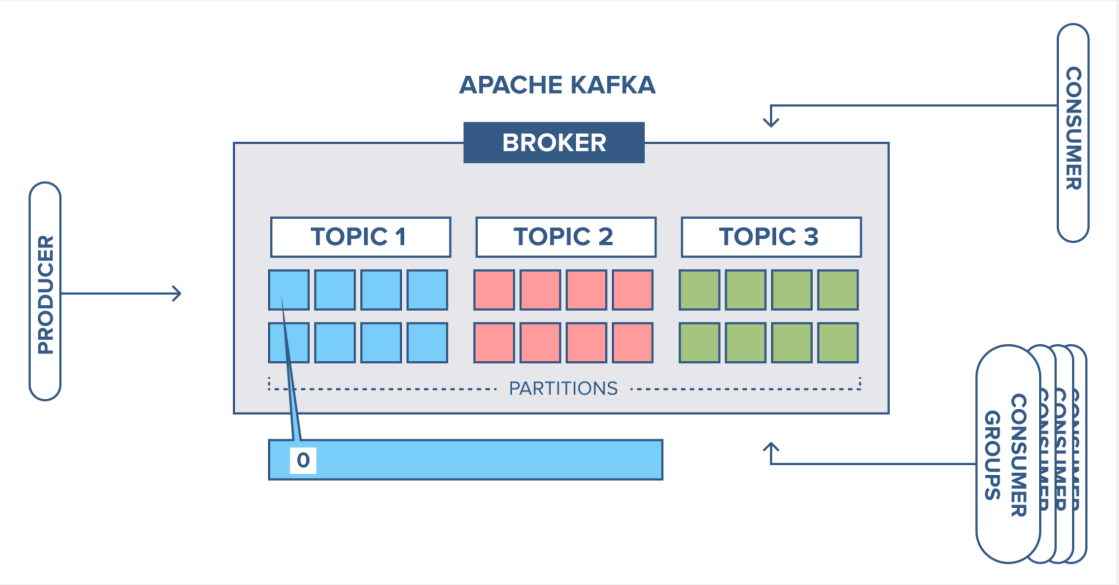
Kafka is great for durable and scalable ingestion of streams of events coming from many producers to many consumers.

Spark is great for processing large amounts of data, including real-time and near-real-time streams of events.

Kafka producer will publish the messages to a topic and the consumer acts as the receiver who will be subscribed to a topic.

DESIGN

- Check the latest version of Kafka
- Start the zookeeper
- Start kafka brokers
- Start kafka topics



STEP I

IMPLEMENTATION

$ bin/kafka-topics.sh –create –topic input_recommend_product –zookeeper localhost:2181 –partitions 3 – replication-factor 1

$ bin/kafka-topics.sh –create –topic input_recommend_product –bootstrap-server localhost:9092 –partitions 3 –replication-factor 1

SPARKPROCESSOR.PY

```python
import sys
from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext

#from pyspark.streaming.kafka import KafkaUtils


def process_events(event):
    return (event[0], Counter(event[1].split(" ")).most_common(3))


def push_back_to_kafka(processed_events):
    list_of_processed_events = processed_events.collect()
    producer.send('output_event', value = str(list_of_processed_events))


def spark_context_creator():
    conf = SparkConf()
    #The master URL to connect and set name for our app
    conf.setMaster("spark://34.70.211.224:7077").setAppName("ConnectingDotsSparkKafkaStreaming")
    sc = None
    try:
        sc.stop()
        sc = SparkContext(conf=conf)
    except:
        sc = SparkContext(conf=conf)
    return sc


if __name__ == "__main__":
    sc = spark_context_creator()
    ssc = StreamingContext(sc, 1)

    kafkaStream = KafkaUtils.createStream(ssc, 'localhost:2181', 'test-consumer-group', {'input_event':1})
    lines = kafkaStream.map(lambda x : process_events(x))

    producer = KafkaProducer(bootstrap_servers='localhost:9092', value_serializer=str.encode, key_serializer=str.encode)
```

IMPLEMENTATION

$ pip3 install msgpack

$ pip3 install kafka-python

CONSUMER.PY

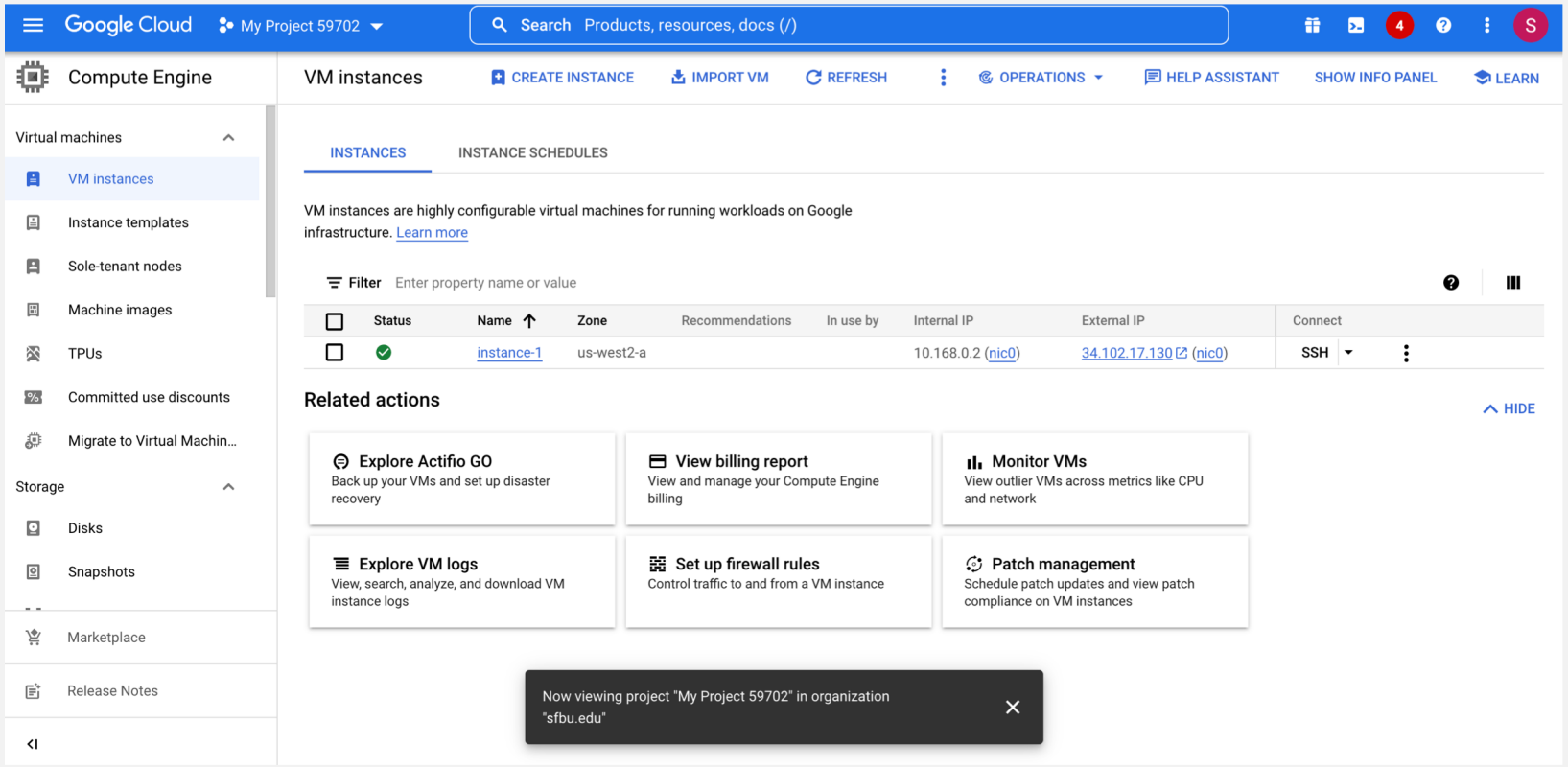
```
1  from kafka import KafkaConsumer
2
3  consumer = KafkaConsumer('input_recommend_product', bootstrap_servers = ['localhost:9092'])
4  for msg in consumer:
5    print(msg)
```

RESULT

```
ConsumerRecord(topic='input_recommend_product', partition=2, offset=0, timestamp=1669973195741, timestamp_type=0, key=No
ne, value=b'(1, Main Menu), (2, Phone) , (3, Smart Phone), (4, iPhone)', headers=[], checksum=None, serialized_key_size=
-1, serialized_value_size=58, serialized_header_size=-1)
```

STEP2

PREREQUISITE - GCP



IMPLEMENTATION

$ wget https://dlcdn.apache.org/spark/spark-3.3.1/spark-3.3.1-bin-hadoop3.tgz

$ tar -xvf spark-3.3.1-bin-hadoop3.tgz

$ ln –s /home/spark-3.3.1-bin-hadoop3 /home/spark

$ export spark_home=/home/spark

$ export PATH=$SPARK_HOME/bin:$PATH

$ export PATH=$SPARK_HOME/sbin:$PATH

$ echo $PATH

IMPLEMENTATION

$ pyspark

$ start-master.sh

$ start-slave.sh spark://34.172.96.149:7077

$ nc –lk 9999

$ ./bin/spark-submit examples/src/main/python/streaming/network_wordcount.py localhost 9999

STEP3

IMPLEMENTATION

$ wget https://dlcdn.apache.org/spark/spark-3.3.1/spark-3.3.1-bin-hadoop3.tgz

$ tar –xvf kafka_2.12-3.3.1.tgz

$ pip3 install msgpack

$ pip3 install kafka-python

$ wget https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8-assembly_2.11/2.3.2/spark-streaming-kafka-0-8-assembly_2.11-2.3.2.jar

$ vi pyspark_script/sparkProcessor.py

SPARKPROCESSOR.PY

```python
import sys
from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext

#from pyspark.streaming.kafka import KafkaUtils


def process_events(event):
    return (event[0], Counter(event[1].split(" ")).most_common(3))


def push_back_to_kafka(processed_events):
    list_of_processed_events = processed_events.collect()
    producer.send('output_event', value = str(list_of_processed_events))


def spark_context_creator():
    conf = SparkConf()
    #The master URL to connect and set name for our app
    conf.setMaster("spark://34.70.211.224:7077").setAppName("ConnectingDotsSparkKafkaStreaming")
    sc = None
    try:
        sc.stop()
        sc = SparkContext(conf=conf)
    except:
        sc = SparkContext(conf=conf)
    return sc


if __name__ == "__main__":
    sc = spark_context_creator()
    ssc = StreamingContext(sc, 1)

    kafkaStream = KafkaUtils.createStream(ssc, 'localhost:2181', 'test-consumer-group', {'input_event':1})
    lines = kafkaStream.map(lambda x : process_events(x))

    producer = KafkaProducer(bootstrap_servers='localhost:9092', value_serializer=str.encode, key_serializer=str.encode)
```

IMPLEMENTATION

$ start-master.sh

$ ./spark/bin/spark-submit –jars myrun/spark-streaming-kafka-0-10_2.12-3.3.1.jar –master spark://34.70.211.224:7077 –deploy-mode client myrun/sparkProcessor.py

ENHANCEMENT IDEAS

- We can explore with processing messages in real time.

CONCLUSION

- We tried to integrate three frameworks.

REFERENCES

- Course mateiral
- Apache Kafka