IERG 4330/ ESTR4316 /IEMS 5730 Spring 2022
Homework 4

Release date: Mar 31, 2022
Due date: 23:59:00, Apr 19, 2022
The solution will be posted right after the deadline, so no late homework will be accepted!

Every Student MUST include the following statement, together with his/her signature in the submitted homework.

I declare that the assignment submitted on Blackboard system is original except for source material explicitly acknowledged, and that thea same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.

Signed (Student_____) Date:_____

Name_____ SID_____

Submission notice:

- Submit your homework via the elearning system

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value and justify any assumptions you make. You will be graded not only on whether your answer is correct but also on whether you have done an intelligent analysis.

Q1A. [10 bonus marks] Multi-node Kafka Cluster Setup

Kafka is a distributed streaming platform used for building scalable, fault-tolerant, and real-time data pipelines. In this question, you are required to set up a multi-node Kafka cluster. You can follow [14] or other tutorials to set up the multi-node Kafka cluster. **In your homework submission, show the key steps and screenshots to verify all the processes are running.**

- You are required to run 2 brokers and each broker with 2 partitions [15] in your Kafka cluster.
- After installing the multi-node Kafka cluster, you need to create a Topic and transmit the message “my test message” from the *kafka-console-consumer* to *kafka-console-producer*. Following commands may be useful:

```
$ bin/kafka-topics.sh --create --zookeeper localhost:2181
--replication-factor 2 --partitions 2 --topic my-test-topic
                                //create a topic
$ bin/kafka-console-producer.sh --broker-list localhost:9092
--topic my-test-topic
                                //publish some messages to our topic
$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic
my-test-topic --from-beginning
                                //consume some messages of our topic
```

Q1B. [Mandatory for ESTR4316] [15 bonus marks for others] Setup Kafka Cluster over Kubernetes

In this question, you are required to integrate Kafka with Spark RDD Streaming over Kubernetes. The overall procedure would involve:

1. Setup a multi-node Kafka cluster (with 3 ZooKeeper pods & 3 Kafka pods) over Kubernetes via Helm.
2. Write and run a Kafka producer in a Kubernetes pod to capture, store, and serve the Twitter dataset [9]. The dataset contains many tweets.
3. Write a Spark RDD streaming program to consume and process the data from Kafka to count the popular hashtags carried by the tweets in the dataset.
4. Submit and run this Spark RDD streaming program in IE DIC Kubernetes or other Kubernetes platforms.

You must use this Kafka cluster over K8s to finish Q2 and Q3 in order to get full marks for this question. In your homework submission, show the key steps and screenshots to verify all the processes are running.

In this part, you are required to set up a multi-node Kafka cluster (with 3 Zookeeper pods & 3 Kafka pods) over Kubernetes via Helm. Helm is a package manager which provides an easy and simple method to find, use, and share software built for Kubernetes. More detailed instructions are as follows:

i) Setup multi-node Kafka cluster via Helm

- Follow the guide [1] to install Helm on your local machine. This local machine should be the same machine running your single-node Spark in HW#3 (i.e., this machine needs to install Docker, Spark, kubectl, and Helm).
- Use the following commands to launch a multi-node Kafka cluster over Kubernetes. The "`kafka-value.yaml`" file[2] is an example configuration file for a Kafka cluster. It already contains 3 Zookeeper pods and 3 Kafka pods running as Kubernetes statefulset without any volume mounted.

```
$ helm repo add incubator https://charts.helm.sh/incubator
$ helm install -f kafka-value.yaml -n [SID] kafka incubator/kafka
```

ii) Verify the Setup of the Kafka Cluster

In order to verify the setup of the Kafka cluster and make sure your Spark program can also connect with Kafka on top of Kubernetes, you are required to launch a simple testclient pod and play some simple Kafka demos within this pod. More detailed instructions are as follows:

- We provide the YAML file "`kafka-testclient.yaml`" [3] for the testclient pod. Launch the pod via the following commands:

```
$ kubectl apply -f kafka-testclient.yaml -n [SID]
$ kubectl exec -it testclient bash -n [SID]
```

- Login into the bash terminal of the testclient pod via "kubectl exec -it testclient -n [SID] bash". Then create a topic and transmit the message "my test message" from "kafka-console-consumer" to "kafka-console-producer" [4]. The following commands may be useful:

```
$ bin/kafka-topics.sh --create --zookeeper kafka-zookeeper:2181
--replication-factor 1 --partitions 1 --topic my-test-topic
$ bin/kafka-console-producer.sh --broker-list kafka-headless:9092
--topic my-test-topic
$ bin/kafka-console-consumer.sh --bootstrap-server kafka:9092
--topic my-test-topic --from-beginning
```

Q2. [50 marks + 5 bonus marks] Count the Most Frequent Hashtags with Spark RDD Streaming

In this question, you are required to count the most frequent words under the Twitter hashtag "bitcoin". Specifically, you are required to write a Kafka producer to ingest the Twitter dataset [9] into Kafka. The Kafka producer should send the data to the Kafka cluster based on the time interval (see below for detailed instructions). You then write a Spark RDD streaming program to consume the data from Kafka (i.e., working as a Kafka consumer) and count the top-30 most frequent hashtags among the tweets. The basic idea is to use Kafka as the pipeline to transmit tweets from the dataset to Spark (i.e., "KafkaUtils.createDirectStream" class worked as a Kafka Consumer). You need to use the low-level streaming interface (Spark RDD Streaming) and launch your Spark program over Kubernetes/ your own Kafka cluster/ IE DIC cluster [16]. If you want to get the bonus marks for Q1B, you must use the K8s cluster in this question. More detailed instructions are as follows:

(a) [50 marks]

i) Create a topic with 2 brokers and 2 partitions for each broker. Then write a Kafka producer or use the command-line tool to ingest the dataset [9] into Kafka. One simple method is as follows:

- (For K8s) Enter the bash terminal of the testclient pod.
- Download the dataset [9].
- Create a topic.

```
$ bin/kafka-topics.sh --create --zookeeper kafka-zookeeper:2181
--replication-factor 2 --partitions 2 --topic <topic>
```

- Write a Kafka producer to read and spread the dataset. Below is the format of the dataset:

```
$ tweet1,timestamp1\n
$ tweet2,timestamp2\n
```

You are required to ingest the data based on the time interval to model the real Twitter streaming scenario. For example, after sending tweet1, the Kafka producer needs to wait for a few seconds (i.e., timestamp2 - timestamp1) before sending the next tweet(i.e., tweet2). We provide a Kafka producer [12] written in python for your reference. You can modify this file or write your own Kafka producer.

ii) Refer to [5] and [6] and write a Spark RDD streaming program to consume the data from Kafka (i.e., working as a Kafka consumer) and determine the top-30 most frequent hashtag for each time-window. You need to use a sliding window [13] of length of 5 minutes with a 2-minute sliding interval. Besides, you should start your Spark program and the Kafka producer at the same time. Submit your Spark jobs to your own Kafka cluster/ your own Kafka over K8s cluster/ IE DIC cluster/ IE DIC Kubernetes cluster.

(b) [5 Bonus marks] Fault-tolerance in Spark Streaming and Kafka

In this part, you need to figure out the fault-tolerance mechanism for Kafka over Kubernetes.

i) Re-run your Spark application from Q2(a), but try to suddenly kill a Spark executor pod using the command line “kubectl delete pod xxxxxx” in the middle of its execution. Describe and explain your observations. You should use “kubectl get pods” to check the status of each pod with timestamps and use “kubectl log Spark-driver-pod” to figure out the fault-tolerance mechanism from logs.

ii) Kill your ZooKeeper/Kafka pods. You should use “kubectl get pods” to check the status of each pod with timestamps and use “kubectl log”. Besides, you need to use the testclient of your Kafka and try to send a message under the same topic as Q1B. Describe and explain your observations.

Q3. [40 marks] Spark Structured Streaming

Use Spark structured streaming [7][10] to perform exactly the same task in Q2(a) using the same dataset. If you want to get the bonus marks for Q1B, you must use the K8s cluster in this question.

Reference

[1] Install Helm: https://helm.sh/docs/intro/install/

[2] Kafka-value:

https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/kafka-value.yml

[3] Test client:

https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/kafka-testclient.yml

[4] Kafka documentation: http://kafka.apache.org/documentation

[5] Spark Streaming programming guide:

https://spark.apache.org/docs/latest/streaming-programming-guide.html

[6] Spark RDD streaming tutorial

https://www.rittmanmead.com/blog/2017/01/getting-started-with-spark-streaming-with-python-and-kafka/

[7] Spark structured streaming guide:

https://spark.apache.org/docs/latest/streaming-programming-guide.html#dataframe-and-sql-operations

[8] Persistent Volumes

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

[9] Twitter dataset

https://www.dropbox.com/s/jdck5tip9v4tzfw/new_tweets.txt?dl=0

[10] Structured Streaming + Kafka Integration Guide

https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

[11] Kafka over Kubernetes with PersistentVolume

https://platform9.com/blog/how-to-set-up-and-run-kafka-on-kubernetes/

[12] Kafka Producer

https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/kafka_producer.py

[13] Spark Streaming Sliding Window

https://spark.apache.org/docs/latest/streaming-programming-guide.html#window-operations

[14] Kafka installation

https://kafka.apache.org/quickstart

[15] Set the number of partition in Kafka
http://kafka.apache.org/documentation.html#brokerconfigs

[16] Kafka over IE DIC

https://mobitec.ie.cuhk.edu.hk/ierg4330/tutorials/09_kafka/