

Assignment #9

Worth: 5 points + 5 points extra credit

Exercise 1) 5 points

Read the article “Real-time stream processing for Big Data” available on the blackboard in the ‘Articles’ section and then answer the following questions:

- a) (1.25 points) What is the Kappa architecture and how does it differ from the lambda architecture?

A Kappa architecture is a stream processing system with a powerful stream processor that can handle data at a much faster rate than it is being incoming. All data is periodically recomputed in lambda architecture by the batch layer, whereas in kappa architecture, the computation is only performed when the business logic is changed by replaying historical data.

- b) (1.25 points) What are the advantages and drawbacks of pure streaming versus micro-batch real-time processing systems?

Advantages:

1. Data retention can be scaled using pure stream-oriented systems because of their low latency and relatively high peritem costs.
2. Latency is reduced by limiting batch size.

Drawbacks:

1. A batch-oriented approach achieves unparalleled resource efficiency at the expense of a high level of latency that is prohibitively high for real-time applications.
2. The Trident process is one-at-a-time, so it favors throughput over latency.

- c) (1.25 points) In few sentences describe the data processing pipeline in Storm.

Topology refers to the storm's transportation system. Spouts are the nodes that consume data and start the data flow. To the downstream nodes known as bolts, which carry out the processing, spouts output tuples. Spouts and bolts are distributed evenly throughout the cluster's nodes via Storm. Storm offers the possibility of at least-once processing through an acknowledgement feature that logs the processing, but it makes no guarantees about the sequence in which tuples are processed.

- d) (1.25 points) How does Spark streaming shift the Spark batch processing approach to work on real-time data streams?

By dividing the stream of incoming data items into manageable batches, converting them into RDDs, and processing them as usual, Spark Streaming adapts Spark's batch-processing technique to real-time requirements. The automatic distribution and flow of data are also taken care of. Before processing by employees, data is ingested and converted into an RDD sequence referred to as a "DStream."

Exercise 2) 5 points (extra credit; if you don't want to try or if you try and can't get things to work, this won't impact your score negatively)

Refer to the python-Kafka Documentation from the Free Books and Chapters section of our blackboard site

#### Step A — Start an EMR cluster

Start up an EMR/Hadoop cluster as previously, but instead of choosing the "Core Hadoop" configuration chose the "Spark" configuration (see below), otherwise proceed as before.

#### Step B – Copy the Kafka software to the EMR master node

Download the kafka\_2.13-3.0.0.tgz file from the blackboard to you PC/MAC. Use the secure copy (scp) program to move this file to the /home/hadoop



```
pradaappss — hadoop@ip-172-31-11-25 -- /kafka_2.13-3.0.0 -- -zsh -- 155x51  
(base) pradaappss@Pradaapps-MacBook-Air ~ % ssh -i /Users/pradaappss/Downloads/emr-key-pair.pem hadoop@ec2-44-211-244-6.compute-1.amazonaws.com  
The authenticity of host 'ec2-44-211-244-6.compute-1.amazonaws.com' (44.211.244.6)' can't be established.  
ED25519 key fingerprint is SHA256:7IhedPaciATtguGSitaeBYKFWilzvB4IsdMyAJRQd.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-44-211-244-6.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```

--| |  
--| (- - - )  
---|N-----| Amazon Linux 2 AMI

```
https://aws.amazon.com/amazon-linux-2/  
32 package(s) needed for security, out of 37 available  
Run "sudo yum update" to apply all updates.  
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
```

```
EEEEEEEEEEEEEEEEEE MMMMMMM      MMMMMMMM RRRRRRRRRRRRRRR  
E:::~::~~::~~::~~::~~::~~:M:::~::~~:M   M:::~::~~:R:::~::~~::~~:  
EE:::~::~~::~~::~~::~~::~~:E M:::~::~~:M   M:::~::~~:R:::~::~~::~~:  
E:::~::~~     EEEE    M:::~::~~:M   M:::~::~~ RR:::~::~~:  R:::~::~~  
E:::~::~~     M:::~::~~M   M:::~::~~M   R:::~::~~  R:::~::~~  
E:::~::~~     M:::~::~~M   M:::~::~~M   M:::~::~~  R:::~::~~RR  
E:::~::~~     M:::~::~~M   M:::~::~~M   M:::~::~~  R:::~::~~RR  
E:::~::~~     EEEE    M:::~::~~M   M:::~::~~M   R:::~::~~RR:::~::~~  
E:::~::~~     EEEE    MM      M:::~::~~M   M:::~::~~M   R:::~::~~R  
EE:::~::~~     E M:::~::~~M       M:::~::~~M   R:::~::~~  R:::~::~~  
E:::~::~~     M:::~::~~M       M:::~::~~M RR:::~::~~  R:::~::~~  
EEEEEEEEEEEEEEEEEE MMMMMMM      MMMMMMMM RRRRRRR      RRRRRR
```

```
[hadoop@ip-172-31-11-25 ~]$ ls  
[hadoop@ip-172-31-11-25 ~]$ ls  
kafka_2.13-3.0.0.tgz  
[hadoop@ip-172-31-11-25 ~]$ tar -xzvf kafka_2.13-3.0.0.tgz  
kafka_2.13-3.0.0/  
kafka_2.13-3.0.0/LICENSE  
kafka_2.13-3.0.0/NOTICE  
kafka_2.13-3.0.0/bin/  
kafka_2.13-3.0.0/bin/kafka-delta-records.sh  
kafka_2.13-3.0.0/bin/trogdor.sh  
kafka_2.13-3.0.0/bin/connect-mirror-maker.sh  
kafka_2.13-3.0.0/bin/kafka-console-consumer.sh  
kafka_2.13-3.0.0/bin/kafka-consumer-perf-test.sh  
kafka_2.13-3.0.0/bin/kafka-log-utils.sh  
kafka_2.13-3.0.0/bin/zookeeper-server-stop.sh  
kafka_2.13-3.0.0/bin/kafka-verifiable-consumer.sh  
kafka_2.13-3.0.0/bin/kafka-features.sh  
kafka_2.13-3.0.0/bin/kafka-acls.sh  
kafka_2.13-3.0.0/bin/zookeeper-server-start.sh  
kafka_2.13-3.0.0/bin/kafka-server-stop.sh  
kafka_2.13-3.0.0/bin/kafka-configs.sh
```

Open up a terminal connection to your EMR master node. Over the course of this exercise, you will need to open up three separate terminal connections to your EMR master node. This is the first, which we will call Kafka-Term:

```
tar -xzf kafka_2.13-3.0.0.tgz
```

Note, this will create a new directory (kafka\_2.13-3.0.0) holding the kafka software release.

```
pip install kafka-python
```

```
pradaapss — hadoop@ip-172-31-11-25:~/kafka_2.13-3.0.0
kafka_2.13-3.0.0/libs/kafka-streams-3.0.0.jar
kafka_2.13-3.0.0/libs/rocksdbjni-6.19.3.jar
kafka_2.13-3.0.0/libs/kafka-streams-scala_2.13-3.0.0.jar
kafka_2.13-3.0.0/libs/kafka-streams-test-utils-3.0.0.jar
kafka_2.13-3.0.0/libs/kafka-streams-examples-3.0.0.jar
[hadoop@ip-172-31-11-25 ~]$ pip install kafka-python
Defaulting to user installation because normal site-packages is not writeable
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
    | 246 kB 35.9 MB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
```

Now enter the following commands into the terminal:

```
cd kafka_2.13-3.0.0
```

```
bin/zookeeper-server-start.sh config/zookeeper.properties &
```

```
bin/kafka-server-start.sh config/server.properties &
```

```
pradaapss — hadoop@ip-172-31-11-25:~/kafka_2.13-3.0.0 — zsh — 155x51
[hadoop@ip-172-31-11-25 ~]$ bin/zookeeper-server-start.sh config/zookeeper.properties &
[1] 28870
[hadoop@ip-172-31-11-25 ~]$ -bash: bin/zookeeper-server-start.sh: No such file or directory

[1]+  Exit 127                  bin/zookeeper-server-start.sh config/zookeeper.properties
[hadoop@ip-172-31-11-25 ~]$ cd kafka_2.13-3.0.0
[hadoop@ip-172-31-11-25 kafka_2.13-3.0.0]$ bin/zookeeper-server-start.sh config/zookeeper.properties &
[1] 1831
[hadoop@ip-172-31-11-25 kafka_2.13-3.0.0]$ [2022-11-08 03:23:25,337] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,338] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,342] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,343] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,343] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,343] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,345] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2022-11-08 03:23:25,345] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2022-11-08 03:23:25,345] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2022-11-08 03:23:25,345] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2022-11-08 03:23:25,348] INFO Log4j 1.2 jmx support found and enabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2022-11-08 03:23:25,358] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,359] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,359] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,359] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,359] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,359] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2022-11-08 03:23:25,359] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2022-11-08 03:23:25,377] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@9f70c54 (org.apache.zookeeper.server.ServerMetrics)
[2022-11-08 03:23:25,381] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistance.FileTxnSnapLog)
[2022-11-08 03:23:25,394] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,395] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,398] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,398] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,398] INFO Server environment:zookeeper.version=3.6.3-6401e4ad2087061bc6b9f80dec2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,398] INFO Server environment:host.name=ip-172-31-11-25.ec2.internal (org.apache.zookeeper.server.ZooKeeperServer)
[2022-11-08 03:23:25,398] INFO Server environment:java.version=1.8.0_342 (org.apache.zookeeper.server.ZooKeeperServer)
```

Remember to end each line with the ampersand (&). This starts up a zookeeper instance and the kafka server. Lots of messages should appear. You might need to tap the return/enter key after messages appear to see the Linux prompt again.

Just leave this terminal window alone after you enter these commands. As you interact with kafka this terminal will display low level diagnostic messages which you can ignore.

#### Step D – Prepare to run Kafka producers and consumers

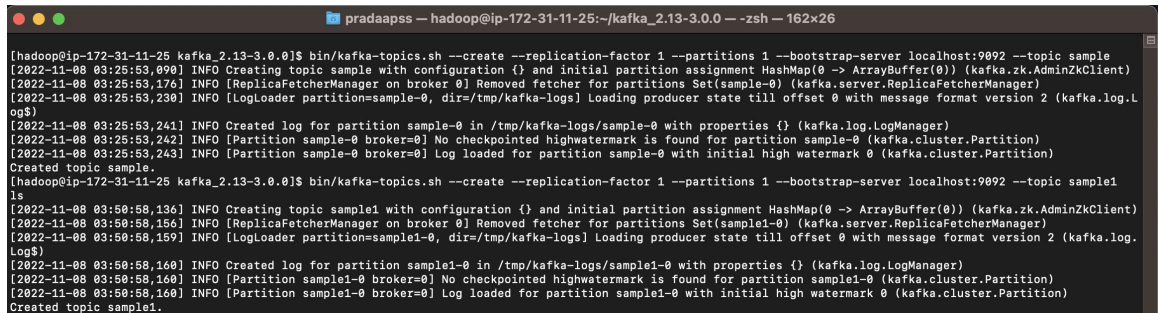
Open a second terminal connection to the EMR master node. Going forward we will call this terminal connection: Producer-Term.

Open a third terminal connection to the EMR master node. Going forward we will call this terminal connection: Consumer-Term.

#### Step E – Create a Kafka topic

In the Producer-Term, enter the following command:

```
cd kafka_2.13-3.0.0  
  
bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --  
bootstrap-server localhost:9092 --topic sample
```



```
pradaapss — hadoop@ip-172-31-11-25:~/kafka_2.13-3.0.0 — zsh — 162x26  
[hadoop@ip-172-31-11-25 kafka_2.13-3.0.0]$ bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server localhost:9092 --topic sample  
[2022-11-08 03:25:53,090] INFO Creating topic sample with configuration {} and initial partition assignment HashMap(0 -> ArrayBuffer(0)) (kafka.zk.AdminZkClient)  
[2022-11-08 03:25:53,176] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(sample-0) (kafka.server.ReplicaFetcherManager)  
[2022-11-08 03:25:53,230] INFO [LogLoader partition=sample-0, dir=/tmp/kafka-logs] Loading producer state till offset 0 with message format version 2 (kafka.log.Log)  
[2022-11-08 03:25:53,241] INFO Created log for partition sample-0 in /tmp/kafka-logs/sample-0 with properties {} (kafka.log.LogManager)  
[2022-11-08 03:25:53,242] INFO [Partition sample-0 broker=0] No checkpointed highwatermark is found for partition sample-0 (kafka.cluster.Partition)  
[2022-11-08 03:25:53,243] INFO [Partition sample-0 broker=0] Log loaded for partition sample-0 with initial high watermark 0 (kafka.cluster.Partition)  
Created topic sample.  
[hadoop@ip-172-31-11-25 kafka_2.13-3.0.0]$ bin/kafka-topics.sh --create --replication-factor 1 --partitions 1 --bootstrap-server localhost:9092 --topic sample1  
[2022-11-08 03:50:58,136] INFO Creating topic sample1 with configuration {} and initial partition assignment HashMap(0 -> ArrayBuffer(0)) (kafka.zk.AdminZkClient)  
[2022-11-08 03:50:58,156] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(sample1-0) (kafka.server.ReplicaFetcherManager)  
[2022-11-08 03:50:58,159] INFO [LogLoader partition=sample1-0, dir=/tmp/kafka-logs] Loading producer state till offset 0 with message format version 2 (kafka.log.Log)  
[2022-11-08 03:50:58,160] INFO Created log for partition sample1-0 in /tmp/kafka-logs/sample1-0 with properties {} (kafka.log.LogManager)  
[2022-11-08 03:50:58,160] INFO [Partition sample1-0 broker=0] No checkpointed highwatermark is found for partition sample1-0 (kafka.cluster.Partition)  
[2022-11-08 03:50:58,160] INFO [Partition sample1-0 broker=0] Log loaded for partition sample1-0 with initial high watermark 0 (kafka.cluster.Partition)  
Created topic sample1.
```

Here we create a new kafka topic called 'sample'. You can use this command to create a topic with any name you like. Try creating a few more topics.

To list the topics that you created you can enter the following into the Producer-Term (note some default topics already exist):

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

a)

In the Producer-Term (or some other way) write a small program, call it 'put.py', using the vi text or some other way of putting a python program onto the EMR

master node. If you like you could use a text editor on your PC/MAC to write the program and then scp it over to your EMR master name.

This program should implement a kafka producer that writes three messages to the topic 'sample'. Recall that you need to convert values and keys to type bytes. The three messages should have keys and values as follows:

Key	Value
'MYID'	Your student id
'MYNAME'	Your name
'MYEYECOLOR'	Your eye color (make it up if you can't remember)

Execute this program in the Producer-Term, use the command line (you might need to provide a full pathname depending on where your python program is such as /home/hadoop/someplace/put.py):

```
python put.py
```

Submit the program as your answer to 'part a' of this exercise.

```
from kafka import KafkaProducer
```

```
from time import sleep
```

```
from json import dumps
```

```
producer = KafkaProducer(bootstrap_servers=['localhost:9092'], value_serializer=lambda x:
```

```
dumps(x).encode('utf-8'), key_serializer=lambda y: dumps(y).encode('utf-8'))
```

```
producer.send('sample', key = 'MyID', value = 'A20512400')
```

```
producer.send('sample', key = 'MyName', value = 'Pradaap S S')
```

```
producer.send('sample', key = 'EyeColor', value = 'Black')
```

```
sleep(5)
```

```
producer.close()
```

```
pradaapss — hadoop@ip-172-31-11-25:~/kafka_2.13-3.0.0 — zsh — 133x47
Last login: Mon Nov 7 21:20:36 on ttys002
(base) pradaapss@Pradaaps-MacBook-Air ~ % ssh -i /Users/pradaapss/Downloads/emr-key-pair.pem hadoop@ec2-44-211-244-6.compute-1.amazonaws.com

--|  _ _ |  )
--| ( _ _ /
---| \ _ _ |
Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
32 package(s) needed for security, out of 37 available
Run "sudo yum update" to apply all updates.
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory

EEEEEEEEEEEEEEEEEEEE MMMMMMM MMMMMMM RRRRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R::::::::::::R
EE::::::::::::::::::::E M::::::::M M::::::::M R::::::::RRRRRR::::R
E::::E EEEEE M::::::::M M::::::::M RR::::R R::::R
E::::E M::::::::M M::M:M::M R::R R::R
E::::EEEEEEEEEE M::M M::M M::M M::M R::RRRRRR::::R
E::::::::::::E M::M M::M M::M M::M R::::::::RR
E::::EEEEEEEEEE M::M M::M M::M R::RRRRRR::::R
E::::E M::M M::M M::M R::R R::R
E::::E EEEEE M::M M M M::M R::R R::R
EE::::::::::::E M::M M::M M::M R::R R::R
E::::::::::::E M::M M::M M::M RR::R R::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM MMMMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-11-25 ~]$ cd kafka_2.13-3.0.0
[hadoop@ip-172-31-11-25 kafka_2.13-3.0.0]$ vim put.py
[hadoop@ip-172-31-11-25 kafka_2.13-3.0.0]$ python put.py
```

b)

In the Consumer-Term, write another small program, call it 'get.py', using the vi text or some other way of putting a python program onto the EMR master node.

This program should implement a kafka consumer that reads the messages you wrote previously from the topic 'sample' and writes them to the terminal.

The output should look something like this:

Key=MYID, Value='your id number'

Key=MYNAME, Value='your name'

Key=MYEYECOLOR, Value='your eye color'

Execute this program in the Consumer-Term. Use the command line:

```
python get.py
```

Note, if needed you can terminate the program by entering 'ctrl-c'.

Submit the program and a screenshot of its output as your answer to 'part b' of this exercise.

```
from kafka import KafkaConsumer
```

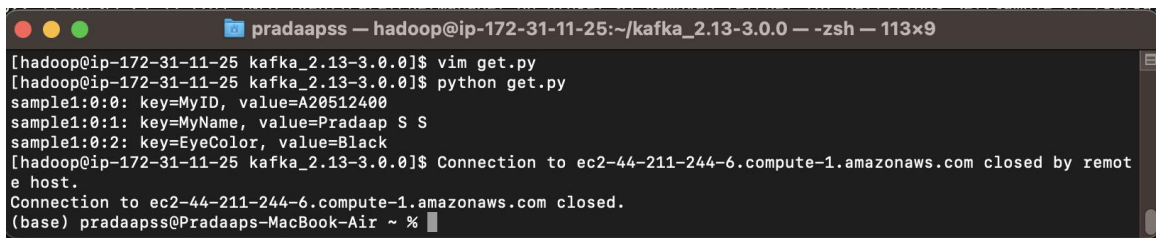
```
from json import loads
```

```
consumer = KafkaConsumer('sample1', auto_offset_reset='earliest',
bootstrap_servers=['localhost:9092'], consumer_timeout_ms=1000, value_deserializer=lambda x:
loads(x.decode('utf-8')), key_deserializer=lambda y: loads(y.decode('utf-8')))

for message in consumer:

    print ("%s:%d:%d: key=%s, value=%s" % (message.topic, message.partition, message.offset,
message.key, message.value))

consumer.close()
```

A terminal window titled 'pradaapss — hadoop@ip-172-31-11-25:~/kafka\_2.13-3.0.0 — zsh — 113x9'. The terminal shows the execution of a Python script 'get.py' using 'python get.py'. The output displays three messages from 'sample1' with their respective keys and values. The first message has key 'MyID' and value 'A20512400'. The second message has key 'MyName' and value 'Pradaap S S'. The third message has key 'EyeColor' and value 'Black'. The terminal also shows connection status messages for 'ec2-44-211-244-6.compute-1.amazonaws.com' and the prompt '(base) pradaapss@Pradaaps-MacBook-Air ~ %'.