CS4830 : Big Data Lab

Lab 7 - Report

Divya K Raman EE15B085 and Sahana Ramnath EE15B109

April 2, 2019

The **python files** used were :

- lab7_gettcp.py : Loads iris.csv from disk and sens each row to “GetTCP” processor in nifi
- lab7_kafka.py : Accepts each row of iris data sent by “PublishKafka” processsor in nifi, predicts its class using the saved trained model ‘finalized_model.sav’ and sends the features and predicted data to “ConsumerKafka” processor in nifi.
- lab7_puttcp.py : Accepts (feature,prediction) rows from PutTCP and print them

These can be found in the submitted folder.

Commands to be run :

- cd kafka_2.1.1-2.10
- bin/zookeeper-server-start.sh config/zookeeper.properties
- bin/kafka-server-start.sh config/server.properties
- bin/kafka-topics.sh --create --zookeeper localhost:9092 --replication-factor 1 --partitions 2 --topic virginicasetosa
- bin/kafka-topics.sh --create --zookeeper localhost:9092 --replication-factor 1 --partitions 2 --topic versicolor
- python lab7_gettcp.py, python lab7_kafka.py, python lab7_puttcp.py (in 3 terminals in parallel)

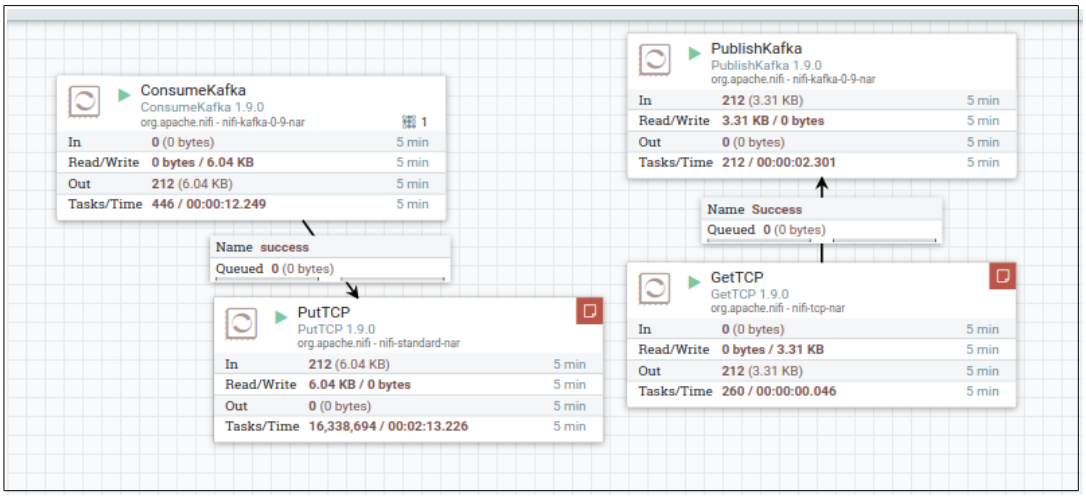
Nifi Architecture :

Node 2 : GetTCP ------> PublisherKafka

Node 1 : ConsumerKafka ------> PutTCP

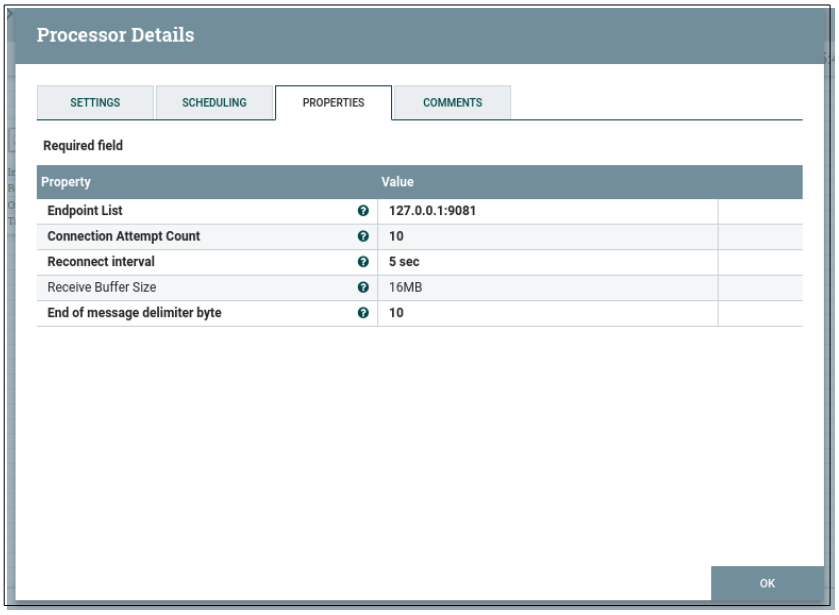
Note : ValidateRecords threw errors and didn’t allow flow of data and hence was omitted in the architecture used. Configuration and architecture are shown in the figures below.

NIFI ARCHITECTURE

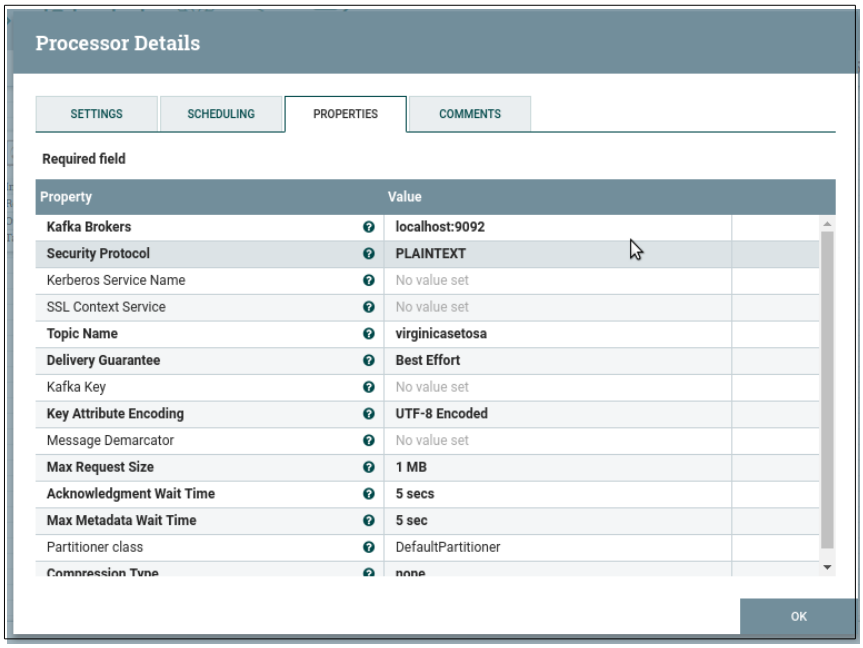


The data flow can be seen in the above figure.

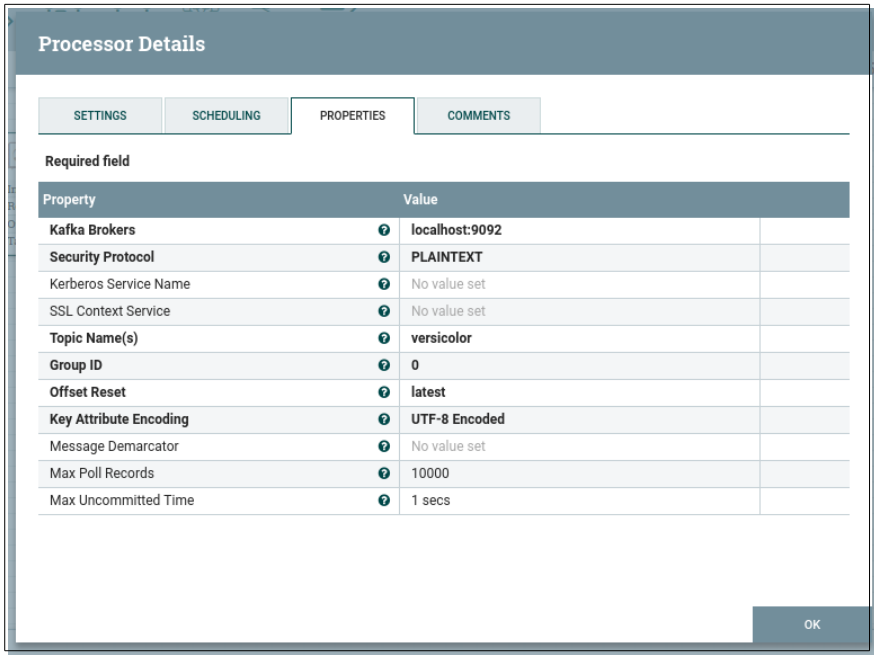
GetTCP Configuration



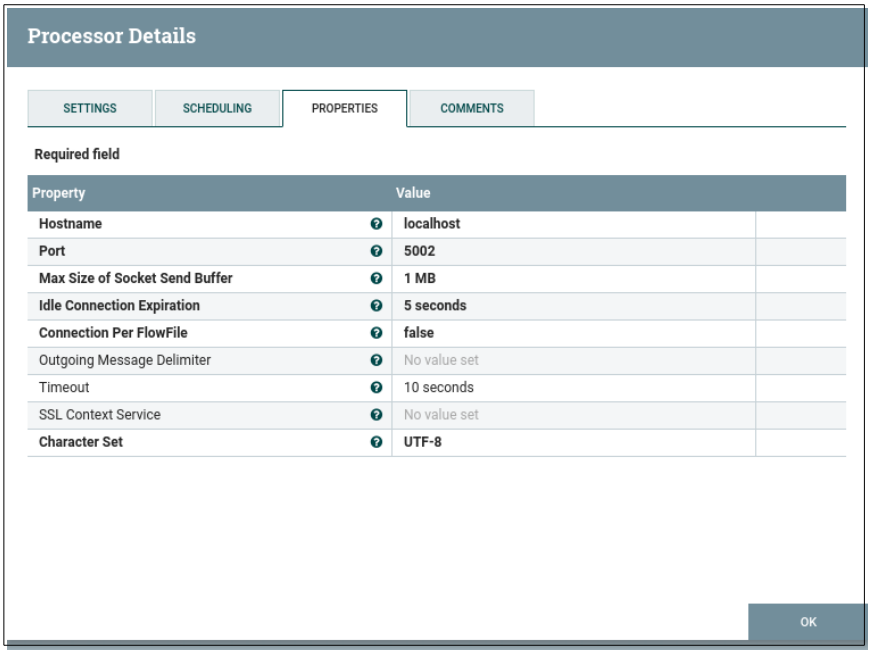
PublisherKafka Configuration



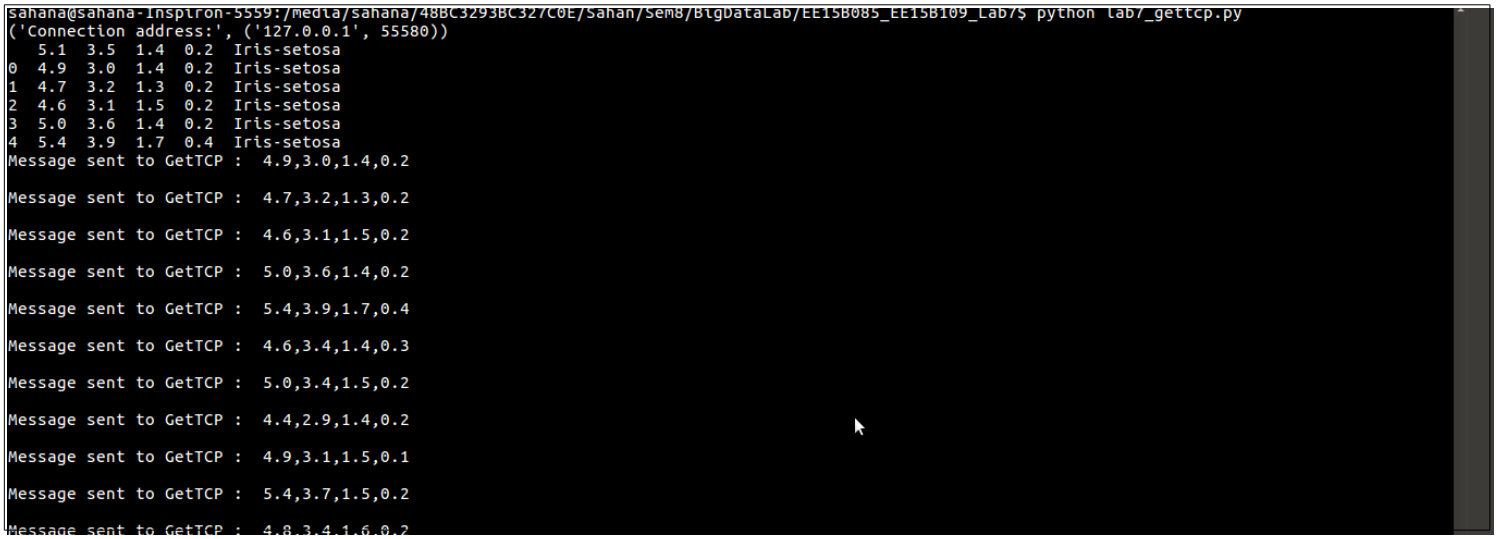
ConsumerKafka Configuration

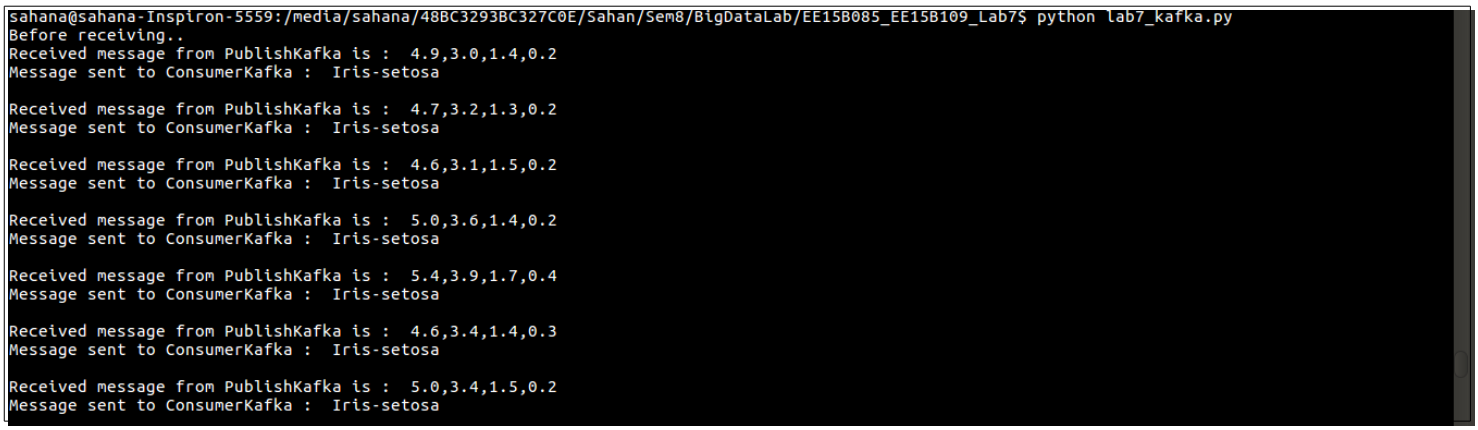


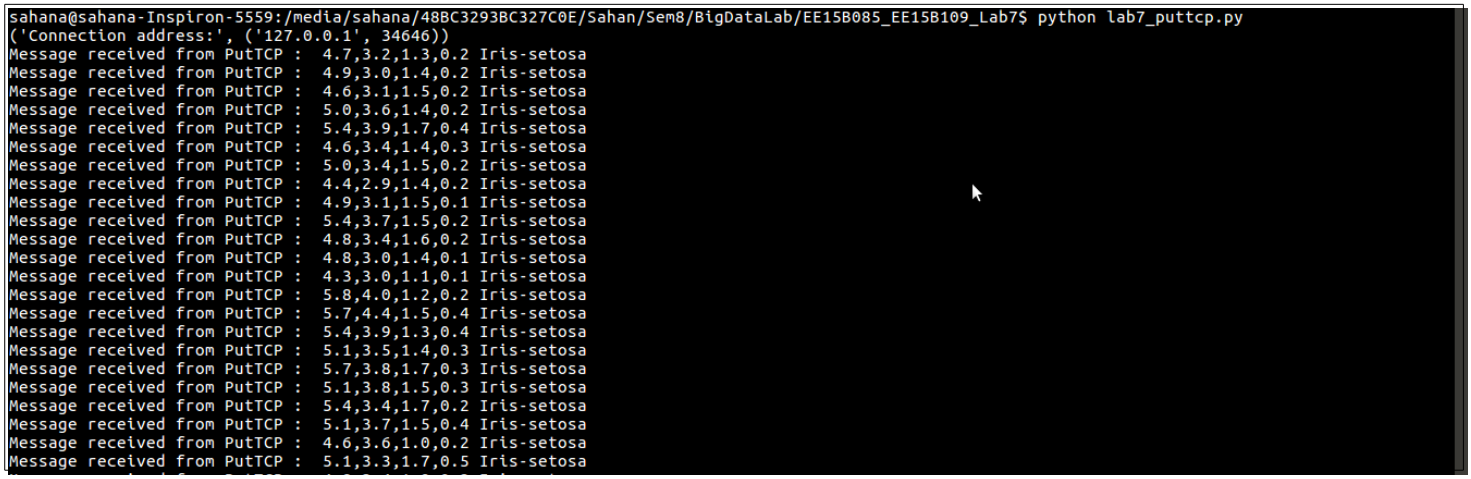
PutTCP Configurartion



SCREENSHOTS OF OUTPUTS







The above three screenshots are by lab7_gettcp.py, lab7_kafka.py and lab7_puttcp.py respectively.

Functions used in Python :

For kafka, KafkaProducer and KafkaConsumer were used.

Functions of module socket were used to send and receive data from nifi through TCP.

OTHER NIFI PROCESSORS

1. GetHDFSFileInfo : Retrieves a listing of files and directories from HDFS. This processor creates a FlowFile(s) that represents the HDFS file/dir with relevant information. Main purpose of this processor to provide functionality similar to HDFS Client, i.e. count, ls, test, etc. This processor is stateless, supports incoming connections and provides information on a dir level.
2. HashAttribute : Hashes together the key/value pairs of several flowfile attributes and adds the hash as a new attribute. Optional properties are to be added such that the name of the property is the name of a flowfile attribute to consider and the value of the property is a regular expression that, if matched by the attribute value, will cause that attribute to be used as part of the hash. If the regular expression contains a capturing group, only the value of the capturing group will be used. For a processor which accepts various attributes and generates a cryptographic hash of each, see “CryptographicHashAttribute”.
3. ListSFTP : Performs a listing of the files residing on an SFTP server. For each file that is found on the remote server, a new FlowFile will be created with the filename attribute set to the name of the file on the remote server. This can then be used in conjunction with FetchSFTP in order to fetch those files.

(This link was referred to for the above).