|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Системы управления данными»

**Практическое занятие № 8**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Алексеева Екатерина Сергеевна, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

1. Практическая работа №8
   1. Цель работы

В этой лабораторной работе мы будем собирать неструктурированные потоковые данные с помощью DStream API.

## Симулируйте источник потоковых данных

### Перейдите в /home/student/Scripts

### Запустите streams.sh сценарий оболочки для начала симуляции источника потоковых данных на сокете Linux на порту 44444

$ /home/student/Scripts/stream\_alice.sh

###### 

## В Jupyter создайте StreamingContext

### Запустите pyspark в локальном режиме с двумя потоками

$ pyspark --master 'local[2]'

### 

### Создайте SparkStreamingContext: В то время как оболочка Spark предоставляет предварительно созданные объекты SparkContext и SparkSession, объект StreamingContext должен быть создан вручную в оболочке

#### Импортируйте библиотеку StreamingContext

from pyspark.streaming import StreamingContext

#### Создайте экземпляр StreamingContext с длительностью DStream 5 секунд

ssc = StreamingContext(sc, 5)

#### Произведите проверку, чтобы убедиться, что ssc был создан правильно

ssc

###### 

## Разработайте логику преобразования для каждого DStream

### Настройте параметры источника потоковых данных

#### Создайте переменную с именем port и установите значение 44444

#### Создайте переменную с именем host и установите значение 'localhost'

host = 'localhost'

port = 44444

### Создайте DStream с помощью socketTextStream. Задайте для хоста и порта переменные, которые мы создали на предыдущем шаге

aliceDS = ssc.socketTextStream(host,port)

### Создайте преобразования в aliceDS, чтобы подсчитать количество слов для каждого DStream

#### Разделите строку на отдельные слова. Подсказка: Используйте flatMap вместо map

#### Создайте кортеж ключ:значение. Каждое слово будет генерировать кортеж (word, 1)

#### Используйте reduceByKey() для объединения всех значений с одним и тем же словом

###### 

### Используйте действие pprint() для быстрой печати wcDS

wcDS.pprint()

### 

### Создайте функцию для печати количества слов для каждого DStream

def printCount(t, r):

print("Word Count at time", t)

for cnt in r.take(10):

print(cnt[0], ":", cnt[1])

### Выполните printCount() для каждого DStream

#### Используйте foreachRDD для выполнения printCount для каждого DStream. foreachRDD передает временную метку и указатель на базовый RDD в текущем DStream.

wcDS.foreachRDD(lambda time, rdd: printCount(time, rdd))

### Запустите механизм потоковой передачи и дождитесь завершения

ssc.start()

ssc.awaitTermination()

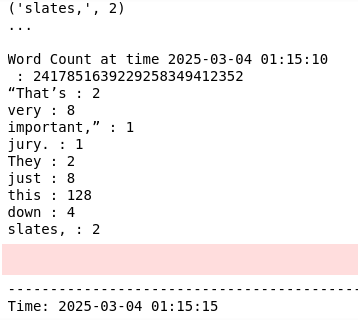
## 

## Протестируйте потоковое приложение

### Как только вы запустите StreamingContext, имитируемый источник потоковых данных подключится и начнет потоковую передачу в указанный сокет

###### 텍스트이(가) 표시된 사진 자동 생성된 설명

### Вы увидите вывод, аналогичный приведенному ниже



# Работа с многопакетным Dstream API

В этой лабораторной работе мы будем работать с несколькими DStream.

## Измените программу и создайте потоковое приложение

### Просмотрите исходные данные

#### Перейдите к /home/student/Data/weblogs

#### Откройте любой из weblog-ов и просмотрите исходные данные.

#### 

62.133.252.161 - 7645 [23/Sep/2021:06:20:05 +0900] "PUT /apps/cart.jsp?appID=7565 HTTP/1.0" 200 4953 "https://www.wallace-briggs.com/app/wp-content/author.html" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.2 (KHTML, like Gecko) Chrome/37.0.899.0 Safari/534.2"

#### 

#### Первым элементом weblog является IP-адрес, через который пользователь подключился к веб-серверу

#### Есть (-), за которым следует другое число. Этот третий элемент – user\_id пользователя, просматривающего веб-страницы. В нашем случае этот идентификатор пользователя является идентификатором автора в нашей таблице авторов в Hive.

#### Остальная часть — это временная метка, операция GET/PUT и другие разные элементы, которые нас пока не интересуют.

### Просмотрите код шаблона

#### Перейдите в /home/student/labs/C6U3

#### 

#### Скопируйте multi\_dstream.py.template в mult\_dstream.py

#### 

#### Вам был предоставлен шаблон для начала программирования мульти пакетного приложения Spark Streaming. Программа ожидает получения имени хоста и номера порта, с которого она будет считывать потоковые данные. В нашем случае мы будем передавать ему "localhost" и "44444"

import sys

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

if \_\_name\_\_ == "\_\_main\_\_":

if len(sys.argv) != 3:

print >> sys.stderr, "Usage: multi\_dstream.py <hostname> <port>"

sys.exit(-1)

# get hostname and port of data source from application arguments

hostname = sys.argv[1]

port = int(sys.argv[2])

# Create a new SparkContext

sc = SparkContext()

# Set log level to ERROR to avoid distracting extra output

sc.setLogLevel("ERROR")

# Create and configure a new Streaming Context

# with a 1 second batch duration

ssc = StreamingContext(sc,1)

# Create a DStream of log data from the server and port specified

logs = ssc.socketTextStream(hostname,port)

# Put you application logic here

ssc.start()

ssc.awaitTermination()

#### Шаблонная логика была выполнена за вас. Был создан SparkContext. Был создан StreamingContext с длительностью DStream 1 секунда.

#### Для источника потоковых данных был установлен socketTextStream с указанным именем хоста и номером порта.

### Измените multi\_dstream.py чтобы добавить логику приложения

#### Используйте редактор для изменения multi\_dstream.py . У вас есть возможность использовать vim или KWrite в качестве редактора в текущей среде.

#### Когда мы работаем с несколькими пакетами DStreams, где требуется состояние, необходимо настроить каталог контрольных точек. Добавьте следующее:

ssc.checkpoint("checkpoint\_dir")

#### Потоковые weblogs сохраняются в переменной logs. Подсчитайте, сколько подключений поступает в течение 5 секунд. Рассчитывайте это количество каждые 2 секунды. Для этого используйте преобразование DStream.countByWindow(windowDuration, slideDuration)

cnt = logs.countByWindow(5,2)

#### Используйте pprint() для печати результата

cnt.pprint()

### 

### Имитация потоковых weblog

#### Перейдите к /home/student/labs/C6U3 на локальном диске

#### Используйте команду cat Linux для проверки stream\_web.sh

$ cat stream\_web.sh

#### Скрипт bash просто вызывает port\_stream.py с именем хоста, номером порта и каталогом источника данных для потоковой передачи weblog-ов в сокет Linux.

#### Запустите stream\_web.sh

$ ./stream\_web.sh

###### 

### Скрипт ответит, сообщив вам, что он ожидает подключения к порту 44444

### Запустите multi\_dstream.py

#### Перед запуском программы мы должны переключить оболочку Spark в режим iPython.

#### Перейдите в свой локальный домашний каталог по адресу /home/student

#### Выберите файл .bashrc и отредактируйте файл, чтобы включить режим ipython

###### 

#### Символ # в начале строки указывает на то, что строка является комментарием. Удалите # комментарий из двух строк настройки для режима ipython. Когда # будет удален, KWrite изменит цвет текста, как показано выше, чтобы указать, что команды будут выполнены. Убедитесь, что включены только настройки ipython. Настройки Jupyter должны оставаться закомментированными с помощью # в начале строки

#### Перейдите обратно в /home/student/C6U3

#### Используйте spark-submit с --master 'local[2]', чтобы убедиться, что существует по крайней мере два потока

spark-submit --master 'local[2]' multi\_dstream.py localhost 44444

#### 

#### Результат будет похож на приведенный ниже:

###### 텍스트, 테이블이(가) 표시된 사진 자동 생성된 설명

## Создайте потоковое приложение, которое сохраняет состояние

### Сохраните старую программу и подготовьтесь к новому коду

#### Скопируйте multi\_stream.py в multi\_stream.py.countbywindow

#### Вернитесь к multi\_dstream.py и отредактируйте программу pyspark

#### Удалите следующие две строки кода

cnt = logs.countByWindow(5,2)

cnt.pprint()

### Подсчитайте количество вхождений author\_id

#### Используйте логику wordcount, которую мы видели много раз, чтобы подсчитать количество вхождений author\_id

#### Используйте map() с помощью String.split(" ") для извлечения из weblog

authorCnt = logs.map(lambda line: line.split(" "))

#### Создайте пару RDD, состоящую из (author\_id, 1)

.map(lambda coll: (coll[2], 1))

#### Используйте reduceByKey для добавления всех вхождений author\_id

.reduceByKey(lambda v1, v2: v1+v1)

### Используйте updateStateByKey() для обновления вхождений author\_id.

#### Создайте функцию, которая будет обновлять счетчик. Ожидается, что функция получит текущее состояние и набор новых вхождений. Если необходимо проверить, является ли состояние None (это означает, что мы впервые видим этот author\_id), он должен вернуть количество новых вхождений. Если есть состояние, оно должно возвращать состояние плюс количество новых вхождений. Функция будет аналогична следующей:

def updateAuthorCount(new\_occurences, curr\_state):

if curr\_state == None: return sum(new\_occurences)

else: return curr\_state + sum(new\_occurences)

#### Преобразуйте authorCnt с помощью updateStateByKey, используя функцию updateAuthorCount

totalAuthorCnt = authorCnt \

.updateStateByKey(lambda new\_occurences, curr\_state: updateAuthorCount(new\_occurences, curr\_state))

#### Используйте pprint() для печати totalAuthorCnt

#### Используйте spark-submit с --master 'local[2]', чтобы убедиться, что существует по крайней мере два потока

spark-submit --master 'local[2]' multi\_dstream.py localhost 44444

#### 

#### Выходные данные будут аналогичны приведенным ниже:

###### 텍스트이(가) 표시된 사진 자동 생성된 설명

### В настоящее время приложение выводит авторов, которые заходили на веб-сайт, и частоту их контактов. Однако здесь нет никакого упорядочения или сортировки. Измените приложение таким образом, чтобы авторы с наиболее частыми посещениями веб-сайта отображались первыми

#### Используйте преобразование map(), для изменения (<author\_id>,<frequency>) на (<frequency >,<author\_id >)

mostFreqAuthor = totalAuthorCnt \

.map(lambda tup:(tup[1], tup[0])) \

#### 

#### Используйте transform() для выполнения преобразования sortByKey RDD. Напомним, что transform — это преобразование-оболочка, которое позволяет вызывать преобразования RDD для DStreams. Установите для параметра sortByKey значение False. В этом случае восходящая сортировка является False, а нисходящая - True.

.transform(lambda rdd: rdd.sortByKey(False)) \

#### Используйте map(), чтобы вернуть пару rdd обратно в (<author\_id>,<frequency>)

.map(lambda tup:(tup[1], tup[0]))

#### Используйте spark-submit с --master 'local[2]', чтобы убедиться, что существует по крайней мере два потока

spark-submit --master 'local[2]' multi\_dstream.py localhost 44444

#### Выходные данные будут аналогичны приведенным ниже:

###### 

import sys

from pyspark import SparkContext

from pyspark.streaming import StreamingContext

def updateAuthorCount(new\_occurences, curr\_state):

if curr\_state == None: return sum(new\_occurences)

else: return curr\_state + sum(new\_occurences)

if \_\_name\_\_ == "\_\_main\_\_":

if len(sys.argv) != 3:

print >> sys.stderr, "Usage: multi\_dstream.py <hostname> <port>"

sys.exit(-1)

# get hostname and port of data source from

# application arguments

hostname = sys.argv[1]

port = int(sys.argv[2])

# Create a new SparkContext

sc = SparkContext()

# Set log level to ERROR to avoid distracting extra output

sc.setLogLevel("ERROR")

# Create and configure a new Streaming Context

# with a 1 second batch duration

ssc = StreamingContext(sc,1)

# Create a DStream of log data from the server

# and port specified

logs = ssc.socketTextStream(hostname,port)

# Put you application logic here

ssc.checkpoint("checkpoint\_dir")

authorCnt = logs \

.map(lambda line: line.split(" ")) \

.map(lambda coll: (coll[2], 1)) \

.reduceByKey(lambda v1, v2: v1+v1)

totalAuthorCnt = authorCnt \

.updateStateByKey(lambda new\_occurences, curr\_state: updateAuthorCount(new\_occurences, curr\_state))

mostFreqAuthor = totalAuthorCnt \

.map(lambda tup:(tup[1], tup[0])) \

.transform(lambda rdd: rdd.sortByKey(False)) \

.map(lambda tup:(tup[1], tup[0]))

mostFreqAuthor.pprint()

ssc.start()

ssc.awaitTermination()

# Работа со структурированным потоковым API

В этой лабораторной работе мы будем работать со структурированными потоковыми данными. Источник потоковой передачи будет создан в виде dataframe, где isStreaming имеет значение true.

## Настройка структурированного источника потоковой передачи

### Скопируйте данные posts из HDFS на локальный диск

#### В более ранней лабораторной работе таблица posts в MySQL была импортирована в HDFS в каталог /user/student/posts. Убедитесь, что данные здесь. Если нет, используйте Sqoop для импорта данных.

#### Используйте hdfs dfs -get для копирования данных из HDFS на локальный диск. Используйте следующую команду:

$ cd /home/student/Data

$ hdfs dfs -get posts

#### Убедитесь, что на вашем локальном диске был создан новый каталог с именем posts

#### Перейдите к сообщениям. Существует несколько разделов данных. Существует также файл \_SUCCESS. Удалите этот файл.

$ cd ./posts

$ rm \_SUCCESS

#### 

#### Просмотрите один из разделов данных. Данные представлены в формате CSV, но строки заголовка нет. Нам нужно будет знать информацию о схеме, чтобы использовать эти данные.

#### 

### 

### Просмотрите схему таблицы постов

#### Откройте MariaDB как пользователь student и используйте "student" в качестве пароля

$ mysql -u student -p

#type student when prompted for password

#### Из MariaDB перейдите в базу данных labs и используйте DESC для получения информации о схеме для таблицы posts

MariaDB [(none)]> USE labs;

MariaDB [(labs)]> DESC posts;

#### Выходные данные будут аналогичны приведенным ниже:

###### 

### Выполните скрипт для создания источника потоковой передачи файлов

#### Выполните stream\_posts.sh для потоковой передачи сообщений в сокет Linux на localhost через порт номер 44444

#### 

## Создайте потоковый dataframe

### Перезапустите PySpark для запуска с Jupyter, изменив .bashrc в домашнем каталоге

### 

### Запустите pyspark как минимум с двумя потоками.

pyspark --master local[2]

### Создайте фрейм данных из источника данных сокета

postsDF = spark.readStream \

.format("socket") \

.option("host", "localhost") \

.option("port", "44444") \

.load()

### 

### Сокеты считывают данные с фиксированной схемой. Все данные содержатся в одном столбце с именем "value". Необходимо использовать выражения столбцов для создания столбцов с типами данных, как указано в описании таблицы Posts в MySQL

from pyspark.sql.functions import \*

aPostDF = postsDF \

.withColumn("id",

split(postsDF.value, ",")[0].cast("integer")) \

.withColumn("author\_id",

split(postsDF.value, ",")[1].cast("integer")) \

.withColumn("title",

split(postsDF.value, ",")[2]) \

.withColumn("description",

split(postsDF.value, ",")[3]) \

.withColumn("content",

split(postsDF.value, ",")[4]) \

.withColumn("date",

split(postsDF.value, ",")[5])

## Изменение и преобразование потоковых DataFrame

### После создания базового потокового фрейма данных используйте преобразования dataframe для форматирования данных по желанию

myOutDF = aPostDF \

.select("author\_id",

aPostDF.title[0:10].alias("Post\_Title"),

"date")

## 

## Запустите механизм структурированной потоковой передачи

### Используйте writeStream с консольным форматом для отображения выходных данных в консоли. Установите режим вывода на добавление и не усекайте выходные данные. Установите триггер для каждого микропакета продолжительностью 2 секунды.

myStream = myOutDF.writeStream \

.format("console") \

.option("truncate","false") \

.outputMode("append") \

.trigger(processingTime="2 seconds") \

.start()

myStream.awaitTermination()

### 

### Выходные данные будут аналогичны приведенным ниже

###### 테이블이(가) 표시된 사진 자동 생성된 설명