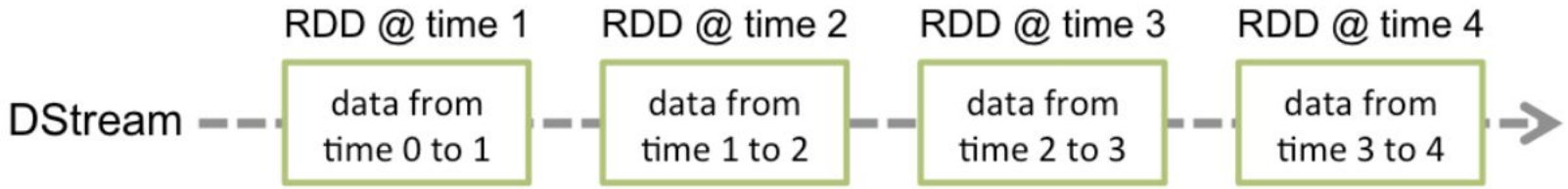
EECS E6893 Big Data Analytics
HW3: Twitter data analysis with Spark Streaming

Tingyu Li, tl2861@columbia.edu

Spark Streaming

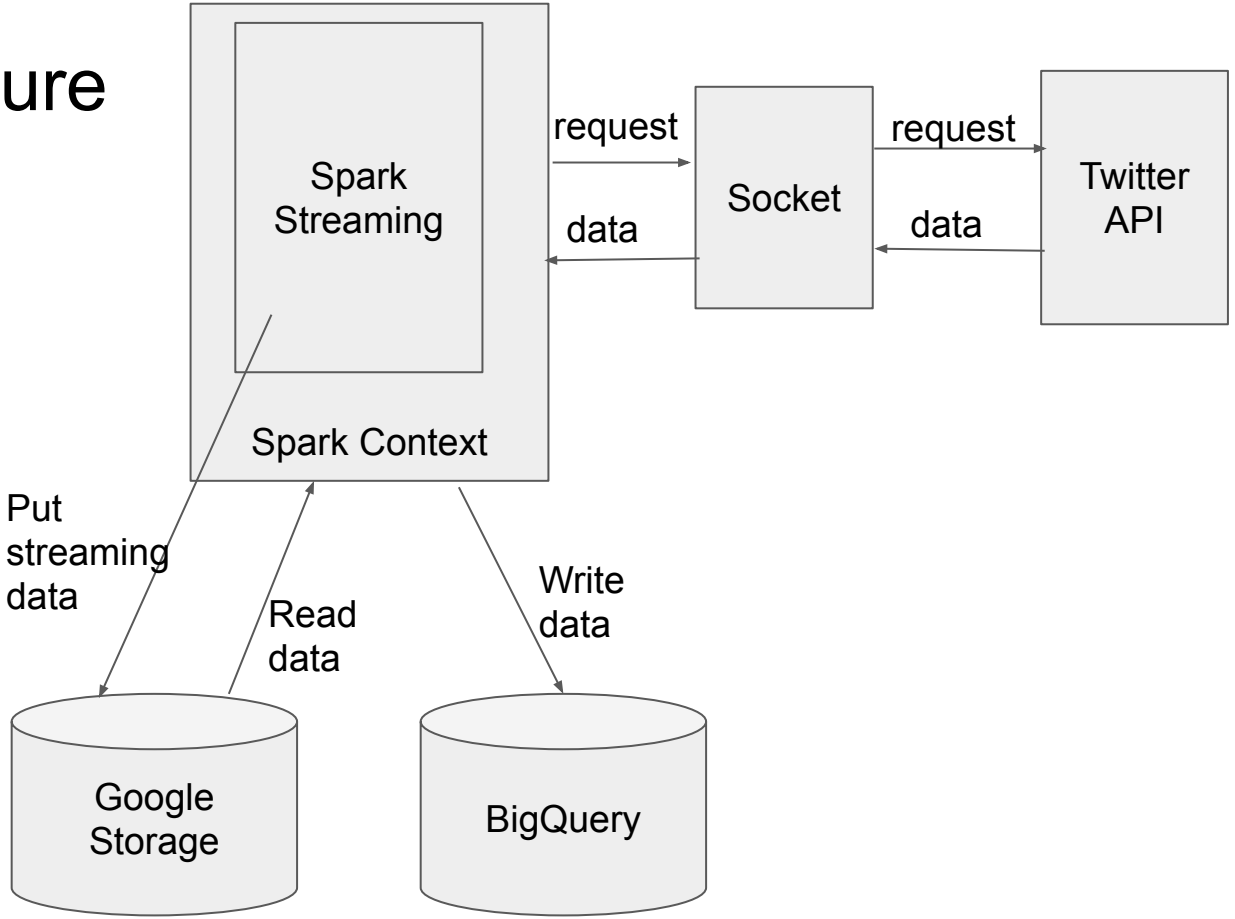
https://spark.apache.org/docs/latest/streaming-programming-guide.html

Dstream



- represents a continuous stream of data
- a continuous series of RDDs

Architecture



Register on Twitter Apps

```
# credentials
# TODO: replace with your own credentials
ACCESS_TOKEN = ''      # your access token
ACCESS_SECRET = ''     # your access token secret
CONSUMER_KEY = ''      # your API key
CONSUMER_SECRET = ''   # your API secret key
```

Socket

Use TCP, need to provide IP and Port for client to connect

```
class twitter_client:
    def __init__(self, TCP_IP, TCP_PORT):
      self.s = s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      self.s.bind((TCP_IP, TCP_PORT))

    def run_client(self, tags):
      try:
        self.s.listen(1)
        while True:
          print("Waiting for TCP connection...")
          conn, addr = self.s.accept()
          print("Connected... Starting getting tweets.")
          sendData(conn,tags)
          conn.close()
      except KeyboardInterrupt:
        exit
```

Spark Streaming

```
if __name__ == '__main__':
    # Spark settings
    conf = SparkConf()
    conf.setMaster('local[2]')
    conf.setAppName("TwitterStreamApp")

    # create spark context with the above configuration
    sc = SparkContext(conf=conf)
    sc.setLogLevel("ERROR")

    # create sql context, used for saving rdd
    sql_context = SQLContext(sc)

    # create the Streaming Context from the above spark context with batch interval size 5 seconds
    ssc = StreamingContext(sc, 5)
    # setting a checkpoint to allow RDD recovery
    ssc.checkpoint("~/checkpoint_TwitterApp")

    # read data from port 9001
    dataStream = ssc.socketTextStream("localhost",9001)
    dataStream.pprint()
```

Create a local StreamingContext with two working thread and batch interval of 5 second.

Create stream from TCP socket IP localhost and Port 9001

Spark Streaming

```
ssc.start()
time.sleep(120)
ssc.stop(stopSparkContext=False, stopGraceFully=True)

# put the temp result in google storage to google BigQuery
saveToBigQuery(sc, output_dataset, output_table_hashtags, output_directory_hashtags)
saveToBigQuery(sc, output_dataset, output_table_wordcount, output_directory_wordcount)
```

Start streaming context

Stop after 120 seconds

Save results to BigQuery

Task1: hashtagCount

```python
def hashtagCount(words):
    """
    Calculate the accumulated hashtags count sum from the beginning of the stream
    and sort it by descending order of the count.
    Ignore case sensitivity when counting the hashtags:
        "#Ab" and "#ab" is considered to be a same hashtag
    You have to:
    1. Filter out the word that is hashtags.
       Hashtag usually start with "#" and followed by a serious of alphanumeric
    2. map (hashtag) to (hashtag, 1)
    3. sum the count of current DStream state and previous state
    4. transform unordered DStream to a ordered Dstream
    Hints:
        you may use regular expression to filter the words
        You can take a look at updateStateByKey and transform transformations
    Args:
        dstream(DStream): stream of real time tweets
    Returns:
        DStream Object with inner structure (hashtag, count)
    """

    # TODO: insert your code here
    pass
```

Task2: wordCount

```
WORD = ['data', 'spark', 'ai', 'movie', 'good']     #the words you should filter and do word count

# Helper functions
```

```
def wordCount(words):
    """
    Calculte the count of 5 sepcial words for every 20 seconds (window no overlap)
    You can choose your own words.
    Your should:
    1. filter the words
    2. count the word during a special window size
    3. add a time related mark to the output of each window, ex: a datetime type
    Hints:
        You can take a look at reduceByKeyAndWindow transformation
        Dstream is a serious of rdd, each RDD in a DStream contains data from a certain interval
        You may want to take a look of transform transformation of DStream when trying to add a time
    Args:
        dstream(DStream): stream of real time tweets
    Returns:
        DStream Object with inner structure (word, (count, timestamp))
    """

    # TODO: insert your code here
    pass
```

Task3: Save results

Create a dataset:

bq mk <Dataset name>

Replace with your own bucket and dataset name:

```
# global variables
bucket = ""    # TODO : replace with your own bucket name
output_directory_hashtags = 'gs://{}/hadoop/tmp/bigquery/pyspark_output/hashtagsCount'.format(bucket)
output_directory_wordcount = 'gs://{}/hadoop/tmp/bigquery/pyspark_output/wordcount'.format(bucket)

# output table and columns name
output_dataset = ''                    #the name of your dataset in BigQuery
output_table_hashtags = 'hashtags'
columns_name_hashtags = ['hashtags', 'count']
output_table_wordcount = 'wordcount'
columns_name_wordcount = ['word', 'count', 'timestamp']
```

Task3: Save results

```
# save hashtags count and word count to google storage
# used to save to google BigQuery
# You should:
#   1. topTags: only save the last DStream
#   2. wordCount: save each DStream
# Hints:
#   1. You can take a look at foreachRDD transformation
#   2. You may want to use helper function saveToStorage
# TODO: insert your code here
```