

UNIVERSIDADE DE BRASILIA

# PROJETO DE PESQUISA

Por Davi Matheus



**Programação de Streaming em clusters**  
**Spark/Kafka**

# Visão Geral

INTRODUÇÃO  
METODOLOGIA  
SOLUÇÕES

CONCLUSÃO



O objetivo desse projeto de pesquisa é que eu possa aprimorar meus conhecimentos de arquitetura Clusters de processamento de fluxo e programação de aplicativos para consumo e Processamento de eventos, em tempo real.



# Introdução

# Problema



**Spark Streaming contabilizando palavras  
de entrada via socket,**

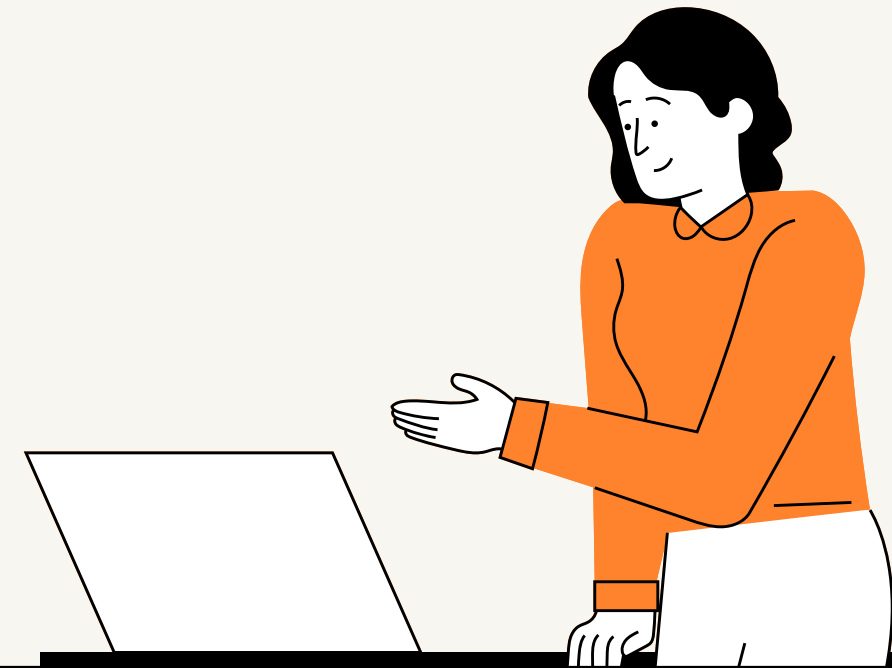
**Spark Streaming contabilizando palavras  
via Apache Kafka.**



# Metodologia



Bom primeiramente, na primeira semana foquei mais na Primeira parte, que seria: **Spark Streaming contabilizando palavras de entrada via socket,**



Entrando na segunda semana foquei mais no entendimento do próprio **Apache Kafka**, buscando em livros e documentações(<https://kafka.apache.org>), na qual eu achei mais difícil do que o normal, já que era a primeira, vez utilizando o kafka.

# Objetivos



Realizar um bom estudo e compreender as tecnologias que envolvem o projeto Apache Kafak e o Apache Spark

Solucionar ambos os problemas  
propostar

Realizar os Graficos através da Bilioteca **Spark Graphx.**

# Fundamentação teórica



## Spark

o Apache Spark, em que ele é um mecanismo multilíngue para executar engenharia de dados, ciência de dados e aprendizado de máquina em máquinas ou clusters de nó único. Foi desenvolvido no AMPLab da Universidade da Califórnia e posteriormente repassado para a Apache Software Foundation que o mantém desde então. Spark provê uma interface para programação de clusters com paralelismo e tolerância a falhas.

## Kafka

Apache Kafka é uma plataforma open-source de processamento de streams desenvolvida pela Apache Software Foundation, escrita em Scala e Java. O projeto tem como objetivo fornecer uma plataforma unificada, de alta capacidade e baixa latência para tratamento de dados em tempo real.

# Soluções



```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode
from pyspark.sql.functions import split
from pyspark.sql.functions import lit
from pyspark.sql.functions import col, upper
```

```
spark = SparkSession \
    .builder \
    .appName("WEB_SOCKET") \
    .getOrCreate()
```

#Criar DataFrame representando o fluxo de linhas de entrada da conexão para localhost:9999





```
lines = spark \
    .readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", 9999) \
    .load()
```



```
18         .load()
19
20 # Divida as linhas em palavras
21 words = lines.select(
22     explode(
23         split(lines.value, " ")
24     ).alias("word")
25 )
26
27 # Gerar contagem de palavras em execução
28 wordCounts = words.groupBy("word").count()
29
30 def foreach_batch_func(df, _):
31     total = df \
32         .groupBy() \
33         .sum() \
34         .select(lit('TOTAL').alias('key'), col('sum(count)').alias('value'))
35
36     df.write.format('console').save()
37     total.write.format('console').save()
38
39 # Comece a executar a consulta que imprime as contagens em execução no console
40 query = wordCounts \
41     .writeStream \
42     .outputMode("complete") \
43     .format("console") \
44     .start()
45
46 query.awaitTermination()
```



111 lines (97 sloc) | 3.17 KB

Raw Blame    

```
1  from pyspark.sql import SparkSession
2  from pyspark.sql.functions import explode
3  from pyspark.sql.functions import split
4  from pyspark.sql.functions import substring
5  from pyspark.sql.functions import window, upper
6  from pyspark.sql.functions import length
7  spark = SparkSession \
8      .builder \
9      .appName("KAFKA") \
10     .getOrCreate()
11
12  # #Criar DataFrame representando o fluxo de linhas de entrada da conexão para localhost:9092 e escrever os toppicos
13
14  lines = spark \
15      .readStream \
16      .format("kafka") \
17      .option("kafka.bootstrap.servers", "localhost:9092") \
18      .option("write", "contador_palavras") \
19      .option('includeTimestamp', 'true') \
20      .load()
21
22  # Divida as linhas em palavras
23  words = lines.select(
24      explode(
25          split(lines.value, "\s+")).alias("word"),
26      lines.timestamp
27  )
28  words = words.select(upper(words.word).alias('word'), words.timestamp)
29
30  # Juntar as palavras
31  wordCounts = words.groupBy("word").count()
32
```

```
30 # Juntar as palavras
31 wordCounts = words.groupBy("word").count()
32
33 # contar o total de palavras lidas
34 total = words \
35     .groupBy() \
36     .count() \
37     .selectExpr("'TOTAL' as key", "CAST(count AS STRING) as value")
38
39 lengths = words \
40     .filter(length(words.word).isin([6, 8, 11])) \
41     .withWatermark("timestamp", "3 seconds") \
42     .groupBy(
43         window(words.timestamp, "3 seconds", "3 seconds"),
44         length(words.word).alias("key")
45     ) \
46     .count() \
47     .selectExpr("CAST(key AS STRING)", "CAST(count AS STRING) as value")
48
49 # Contar 6, 8 and 11
50 lengths = words \
51     .filter(length(words.word).isin([6, 8, 11])) \
52     .withWatermark("timestamp", "3 seconds") \
53     .groupBy(
54         window(words.timestamp, "3 seconds", "3 seconds"),
55         length(words.word).alias("key")
56     ) \
57     .count() \
58     .selectExpr("CAST(key AS STRING)", "CAST(count AS STRING) as value")
59
60
61
62 # Contar as palavras S, P and R
63 letters = words \
64     .filter(upper(substring(words.word, 0, 1)).isin(["S", "P", "R"])) \
65     .withWatermark("timestamp", "3 seconds") \
66     .groupBy(
67         window(words.timestamp, "3 seconds", "3 seconds"),
68         upper(substring(words.word, 0, 1)).alias("key"),
69     ) \
70     .count() \
71     .selectExpr("key", "CAST(count AS STRING) as value")
```

```
72 # Sinks
73 qW = wordCounts \
74     .writeStream \
75     .outputMode("complete") \
76     .format("console") \
77     .start()
78
79 qT = total \
80     .writeStream \
81     .outputMode("complete") \
82     .format("kafka") \
83     .option("kafka.bootstrap.servers", "localhost:9092") \
84     .option('topic', "topics") \
85     .option('checkpointLocation', '/tmp/spark/total-stats') \
86     .start()
87
88 qLen = lengths \
89     .writeStream \
90     .outputMode("update") \
91     .format("kafka") \
92     .option("kafka.bootstrap.servers", "localhost:9092") \
93     .option('topic', "topics") \
94     .option('checkpointLocation', '/tmp/spark/len-stats') \
95     .trigger(processingTime='3 seconds') \
96     .start()
97
98 qLet = letters \
99     .writeStream \
100     .outputMode("update") \
101     .format("kafka") \
102     .option("kafka.bootstrap.servers", "localhost:9092") \
103     .option('topic', "topics") \
104     .option('checkpointLocation', '/tmp/spark/let-stats') \
105     .trigger(processingTime='3 seconds') \
106     .start()
107
108 qLen.awaitTermination()
109 qLet.awaitTermination()
110 qT.awaitTermination()
111 qW.awaitTermination()
```

# Considerações Finais







# Conclusão





**OBRIGADO!**