# Streaming de Dados em Tempo Real: Aula 4

*Prof. Felipe Timbó*

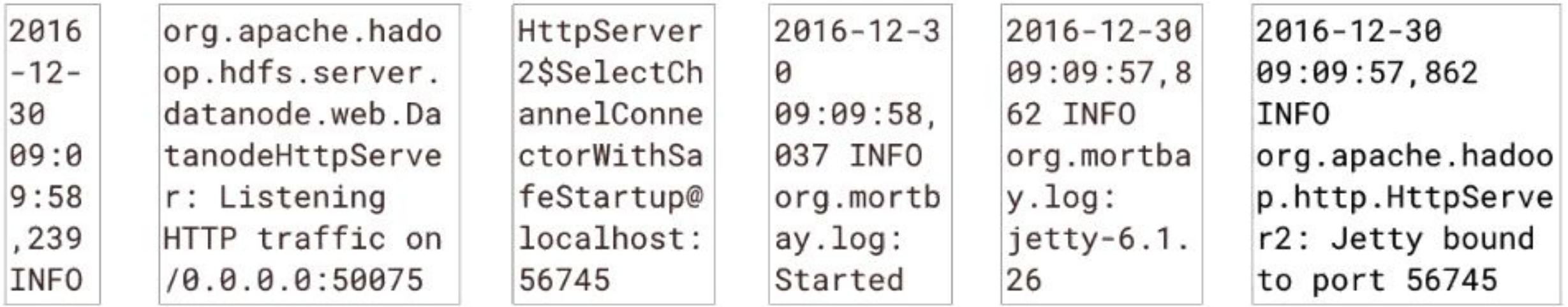
UNI7

# Ementa (dia 4)

- Resolução de problemas com Spark Streaming

# Spark Streaming

# Spark Streaming

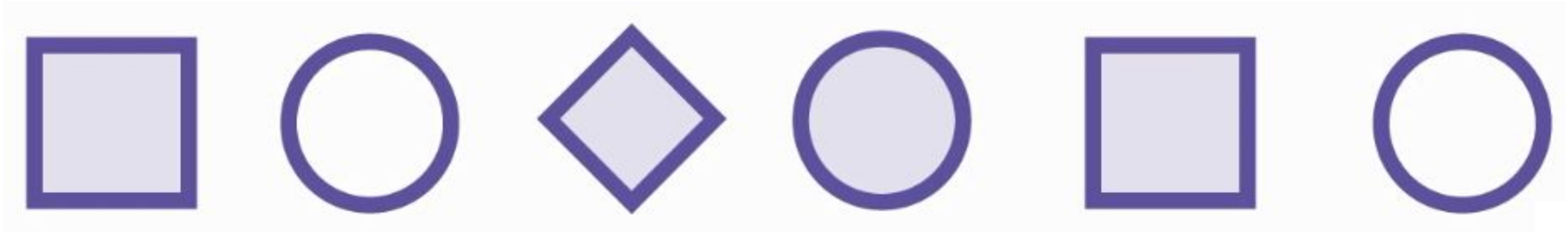



Cada mensagem é uma **entidade** no streaming.

Spark trabalha com streaming de dados usando a mesma abstração do RDD.

# DStream - Discretized Stream

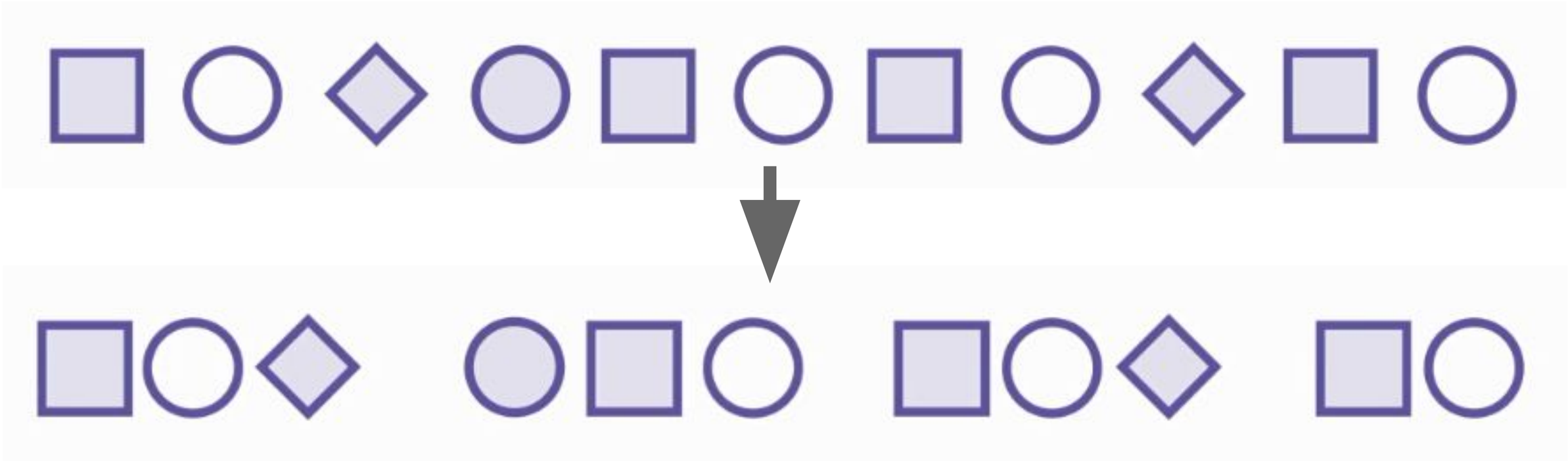
| | | | | | |
|---|---|---|---|---|---|
| 2016-12-30 09:09:58,239 INFO | org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServer: Listening HTTP traffic on /0.0.0.0:50075 | HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:56745 | 2016-12-30 09:09:58,037 INFO org.mortbay.log: Started | 2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26 | 2016-12-30 09:09:57,862 INFO org.apache.hadoop.http.HttpServer2: Jetty bound to port 56745 |



Stream “discretizado” = DStream = Sequência de RDDs

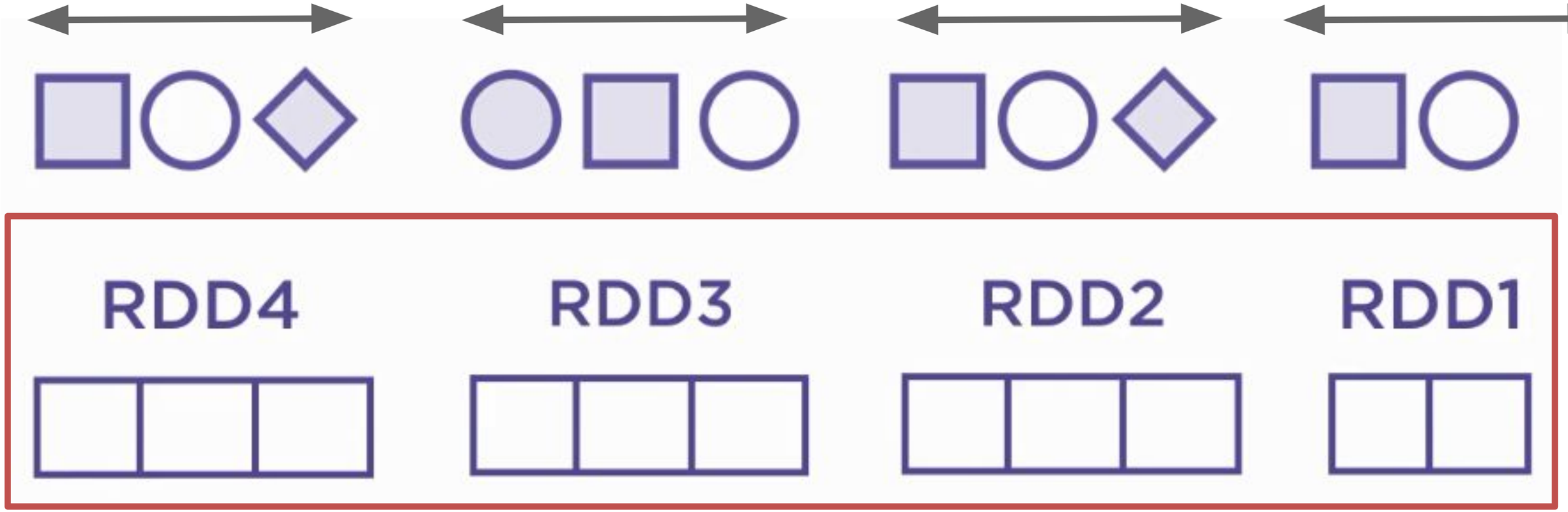
# DStream - Discretized Stream

- Entidades são agrupadas em **batches.** Cada batch é um **RDD**.



# DStream - Discretized Stream

- **Batches** são formados com base em um intervalo de tempo.

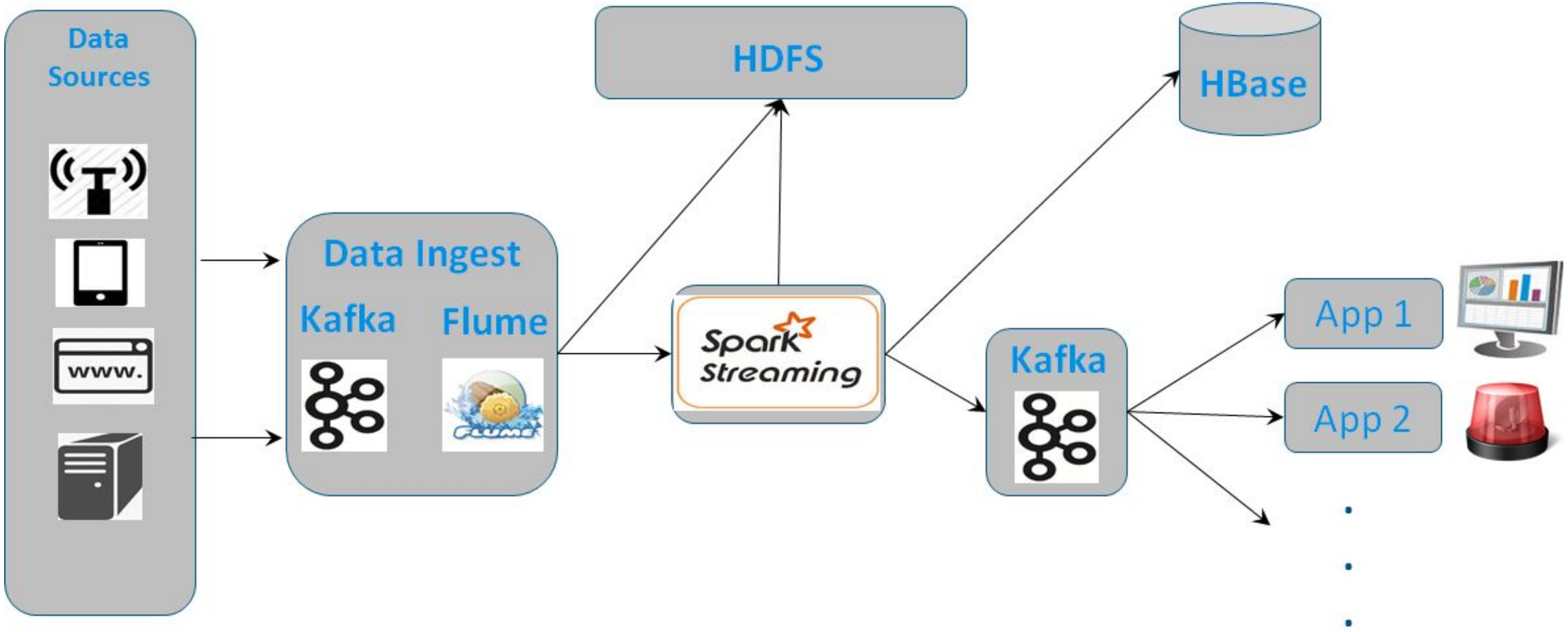


# Stateless vs Stateful


**Transformações são aplicadas em um único RDD**

**Transformações são acumuladas em múltiplos RDD**

# Arquitetura do Processamento de Streaming de Dados



# Spark Streaming - Prática

# Filtragem

```
2016-12-30 09:09:57,862 INFO
org.apache.hadoop.http.HttpServer2: Jetty bound to port
56745
2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26
2016-12-30 09:09:58,037 INFO org.mortbay.log: Started
HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:
56745
2016-12-30 09:09:58,124 INFO
org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServe
r: Listening HTTP traffic on /0.0.0.0:50075
2016-12-30 09:09:58,239 INFO
```

Nesse log, aparece a string **WARN**?

# Filtragem

Terminal 1

1. Baixar e examinar o código no VS Code
   - ➢ `wget www.lia.ufc.br/~timbo/streaming/sparkStreaming.py`
2. Acessar e editar propriedades de log do arquivo log4j
   - ➢ `cd /opt/spark/conf`
   - ➢ `cp log4j.properties.template log4j.properties`
   - ➢ `gedit log4j.properties`
3. Alterar a linha:

`log4j.rootCategory=INFO -> log4j.rootCategory=ERROR`

# Filtragem

Terminal 2

## 4. Rodar o comando netcat

Netcat é uma ferramenta versátil para testes de rede o qual permite ler e escrever dados através das conexões, usando o protocolo TCP/IP.

- `nc -lk 9999`

# Filtragem

VS Code

5. Executar sparkStreaming.py
   - `spark-submit sparkStreaming.py localhost 9999`

# Exercício


1. Alterar o intervalo de tempo para 10 segundos
2. Alterar a palavra buscada para "ERROR"
3. Filtrar apenas por palavras que começam com a letra A (maiúsculo).
4. Filtrar apenas por palavras que terminam com um número qualquer. ex. qwe4, des11, cvb0

# Sumário de Contagens

Acumular a contagem de palavras ao longo do tempo, ou seja, à medida que os dados de streaming vão chegando.

`updateStateByKey`

# Sumário de Contagens

1. Baixar e examinar o código

Terminal 1

```
wget
www.lia.ufc.br/~timbo/streaming/sparkStreaming2.py
```

2. Rodar o comando netcat

Terminal 2

   - `nc -lk 9999`

3. Executar sparkStreaming2.py

Terminal 1

   - `spark-submit sparkStreaming2.py localhost 9999`

# Exercício


- Sumarizar não mais a palavra e quantas vezes ela aparece, mas sim, a quantidade de palavras que existem com um determinado número de letras.

  Exemplo:

  Entrada: wer, dfgt, esdr, asd, bhju, hjyui

  Saída esperada:

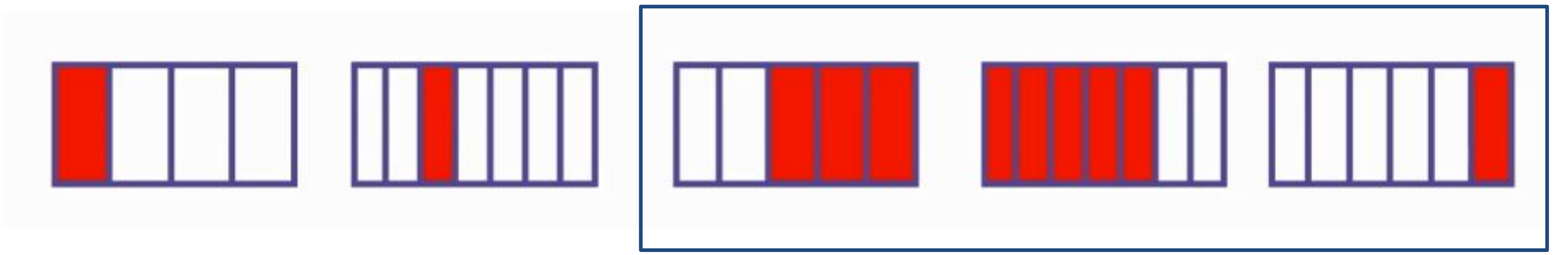
  3: 2  // (2 palavras com 3 letras)

  4: 3  // (3 palavras com 4 letras)

  5: 1  // (1 palavra com 5 letras)

# Janelas

```
2016-12-30 09:09:57,862 INFO
org.apache.hadoop.http.HttpServer2: Jetty bound to port
56745
2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26
2016-12-30 09:09:58,037 INFO org.mortbay.log: Started
HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:
56745
2016-12-30 09:09:58,124 INFO
org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServe
r: Listening HTTP traffic on /0.0.0.0:50075
2016-12-30 09:09:58,239 INFO
```



E se eu quiser o count de strings apenas dos últimos 10 minutos?

# Janelas

1. Baixar e examinar o código

Terminal 1

```
wget
www.lia.ufc.br/~timbo/streaming/sparkStreaming3.py
```

2. Rodar o comando netcat

Terminal 2

   - `nc -lk 9999`

3. Executar sparkStreaming3.py

Terminal 1

   - `spark-submit sparkStreaming3.py localhost 9999`

# Exercício


1. Alterar o tamanho da janela para 15 segundos
   e o intervalo de print dos dados para 3seg.
2. Mostre na janela apenas palavras que possuem "a" (em qualquer posição)
3. Utilizando janela, calcular o número de vezes
   que a palavra ERROR aparece nos últimos 20seg.

# reduceByWindow

1. Baixar e examinar o código

Terminal 1

```
wget
www.lia.ufc.br/~timbo/streaming/sparkStreaming4.py
```

2. Rodar o comando netcat

Terminal 2

   - `nc -lk 9999`

3. Executar sparkStreaming3.py

Terminal 1

   - `spark-submit sparkStreaming4.py localhost 9999`

# Exercício


- Utilizando reduceByWindow, calcular a soma dos caracteres de uma frase inserida via netstat, de forma cumulativa (ao invés da soma de números), no intervalo de 20 segundos

# reduceByKeyAndWindow

1. Baixar e examinar o código

Terminal 1

```
wget
www.lia.ufc.br/~timbo/streaming/sparkStreaming5.py
```

2. Rodar o comando netcat

Terminal 2

   - `nc -lk 9999`

3. Executar sparkStreaming3.py

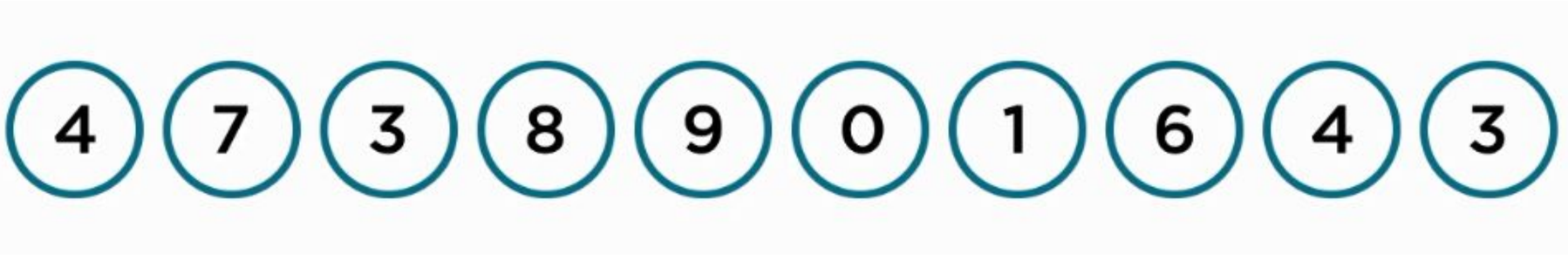
Terminal 1

   - `spark-submit sparkStreaming5.py localhost 9999`

# Exercício


- Filtre registros das palavras WARN, INFO e ERROR, provenientes de streaming (netstat) acumulando-os nos últimos 20 segundos.

# Exercício





Calcule a **soma** dos números que estão chegando via streaming (sem janela).

# Exercício


Utilizando janela e o código Kafka utilizado na aula 2, calcular o número de vezes que aparecem registros com IMC superior a 35, considerando apenas os últimos 20seg.

# Dados de Streaming Estruturados

# Dados de streaming

1. Baixar os dados de streaming:
   - ➢ `wget www.lia.ufc.br/~timbo/streaming/activity-data.zip`

2. Descompactar a pasta

3. Abrir o pyspark
   - ➢ `pyspark`

   - ➢ `from pyspark.sql import SparkSession`
   - ➢ `spark = SparkSession.builder.appName("MyApp").getOrCreate()`

# Dados de streaming estruturados

Carregar os dados de streaming:

```
static = spark.read.json("activity-data/")
```

Mostrar o esquema:

```
dataSchema = static.schema
```

```
root
|-- Arrival_Time: long (nullable = true)
|-- Creation_Time: long (nullable = true)
|-- Device: string (nullable = true)
|-- Index: long (nullable = true)
|-- Model: string (nullable = true)
|-- User: string (nullable = true)
|-- _corrupt_record: string (nullable = true)
|-- gt: string (nullable = true)
|-- x: double (nullable = true)
|-- y: double (nullable = true)
|-- z: double (nullable = true)
```

# Dados de streaming estruturados

Exemplos dos dados:

```
+-------------+-------------------+--------+-----+------+----+--------+-----+-----
| Arrival_Time| Creation_Time     | Device |Index| Model|User|_c...ord|. gt | x
|1424696634224|1424696632226236 85|nexus4_1| 62  |nexus4| a  |    null|stand|-0...
|1424696660715|1424696658723817 26|nexus4_1| 2342|nexus4| a  |    null|stand|-0...
+-------------+-------------------+--------+-----+------+----+--------+-----+-----
```

*gt: que atividade o usuário estava fazendo naquele instante

# Dados de streaming estruturados

Lendo o streaming:

- ```
  streaming =
  spark.readStream.schema(dataSchema).option("maxFilesPer
  Trigger", 1).json("activity-data")
  ```

*maxFilesPerTrigger: controlar a velocidade com que o Spark irá ler todos os arquivos da pasta (não muito usual na prática).
  ```

# Dados de streaming estruturados

Manipulando o streaming:

- `activityCounts = streaming.groupBy("gt").count()`

Evitando a criação de muitas partições:

- `spark.conf.set("spark.sql.shuffle.partitions", 5)`

# Dados de streaming estruturados

Definir como será o output do dado:

- ```
  activityQuery =
  activityCounts.writeStream.queryName("activity_counts").
  format("memory").outputMode("complete").start()
  ```
- ```
  activityQuery.awaitTermination()
  ```

```
from time import sleep
for x in range(10):
    spark.sql("SELECT * FROM activity_counts").show()
    sleep(5)
```

*memory: irá guardar todo o dado em memória (por simplicidade, neste exemplo)

*complete: reescreve todas as chaves e suas contagens a cada trigger

# Exercício

Rodar o mesmo exemplo em formato de script Python

# Seleções e Filtros

```
from pyspark.sql.functions import expr

simpleTransform = streaming.withColumn("stairs",
expr("gt like '%stairs%'"))\
.where("stairs")\
.where("gt is not null")\
.select("gt", "model", "arrival_time", "creation_time")\
.writeStream\
.queryName("simple_transform")\
.format("memory")\
.outputMode("append")\
.start()
```

*append: novos resultados são adicionados à resposta

# Exercício


1. Filtrar apenas pelo gt = bike
2. Filtrar pelo gt = walk ou gt = stand
3. Mostrar apenas as colunas gt, index, model e user

# Agregações

```
deviceModelStats = streaming.cube("gt", "model").avg()\
.drop("avg(Arrival_time)")\
.drop("avg(Creation_Time)")\
.drop("avg(Index)")\
.writeStream.queryName("device_counts").format("memory")\
.outputMode("complete")\
.start()
```

*mais spark aggregations em: https://mungingdata.com/apache-spark/aggregations/

# Exercício


1. Renomear as colunas avg(x), avg(y), avg(z)
2. Mostrar os resultados das médias com apenas duas casas decimais

# Spark Streaming & KAFKA

# Integração com o KAFKA

1. Subir o KAFKA (na pasta kafka)
   - ➢ `bin/zookeeper-server-start.sh config/zookeeper.properties`
   - ➢ `bin/kafka-server-start.sh config/server.properties`

2. Criar um tópico chamado *wctopic*
   - ➢ `bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --topic wctopic`

3. Baixar o arquivo spark_kafka
   - ➢ `cd ~`
   - ➢ `wget www.lia.ufc.br/~timbo/streaming/spark_kafka.py`

# Integração com o KAFKA

4. Rodar o script

   ➢ `spark-submit --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.4.6 spark_kafka.py localhost:9092 wctopic`

5. Criar um broker

   ➢ `bin/kafka-console-producer.sh --broker-list localhost:9092 --topic wctopic`

6. Enviar mensagens

# Exercícios


1. Crie um código em python/spark que leia
   2 tópicos: fortaleza, ceara. Caso a mensagem enviada pelo Broker seja do tópico Fortaleza, imprima a mensagem com o F no início. Caso seja do tópico Ceará, imprima a mensagem com o C no início.
2. Insira uma janela de tempo de 20 segundos
   no exemplo anterior
3. Crie um terceiro tópico denominado wc30warntopic, que
   irá contar apenas as palavras WARN dos últimos 30 seg.