Processamento de Dados em Tempo Real (Streaming) Aula 3

Centro Universitário Christus- UNICHRISTUS

Especialização em Ciência de Dados e Inteligência de Negócios (Big Data e BI)

Prof. Dr. Manoel Ribeiro

Unichristus
Centro Universitário Christus

Conteúdo da disciplina

- Dia 1 (sexta 18:00 às 22:00h)
  - Apresentação
  - Aula motivacional - Qual a importância do processamento de dados em tempo real?
- Dia 2 (sábado 8:00 às 18:00)
  - Fundamentos de sistema de processamento de tempo real
  - Fundamentos da arquitetura Apache Kafka, Apache ZooKeeper
  - Prática com Apache Kafka (tarde)

Conteúdo da disciplina

- Dia 3 (sexta 18:00 às 22:00h)
  - Fundamentos Apache Flume
  - Fundamentos Spark Streaming
  - Configuração Hadoop e Spark
- Dia 4 (sábado 8:00 às 18:00)
  - Fundamentos Introdução ao Apache Storm
  - Implementação de projeto prático/aplicado envolvendo Real Time Analytics
  - Avaliação

Repositório

https://github.com/antoniomralmeida/streaming

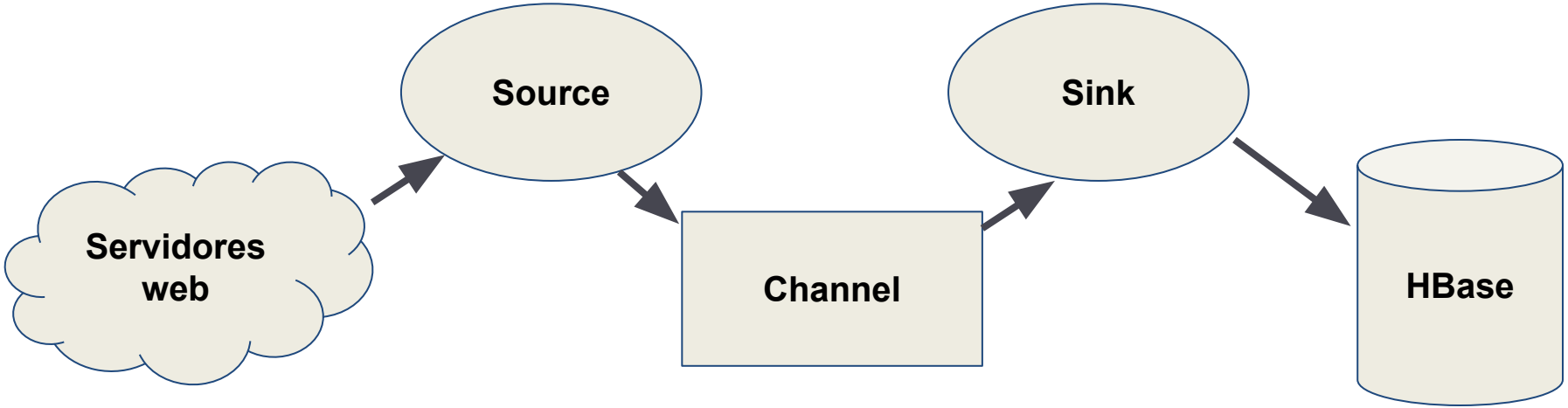
Apache Flume

Apache Flume

- Outra maneira de lidar com streaming de dados em um cluster.
- Feito para inicializar juntamente com o Hadoop.
- Originalmente projeto para lidar com agregação de log.

Flume: Agentes

- Agentes: source, channel, sink.



Flume: Agentes

- Source
  - De onde o dado vem
  - Podem haver *Selectors* e *Interceptors*
- Channel
  - Como o dado é transferido (memória ou arquivo)
- Sink
  - Para onde o dado irá seguir
  - Pode se conectar a apenas a um canal
    - Um canal deleta uma mensagem quando o sink a processa

Source: Tipos de conexões

- Spooldir (diretório)
- Avro (formato específico do hadoop)
- Kafka
- Exec (saída de um prompt de comando no Linux)
- Thrift (semelhante ao Avro)
- Netcat (qualquer porta TCP)
- HTTP
- Customizável

Sink: Tipos de conexões

- HDFS
- Hive
- Avro
- Thrift
- Elasticsearch
- Kafka
- Customizável

Flume: buffer entre os dados e o cluster

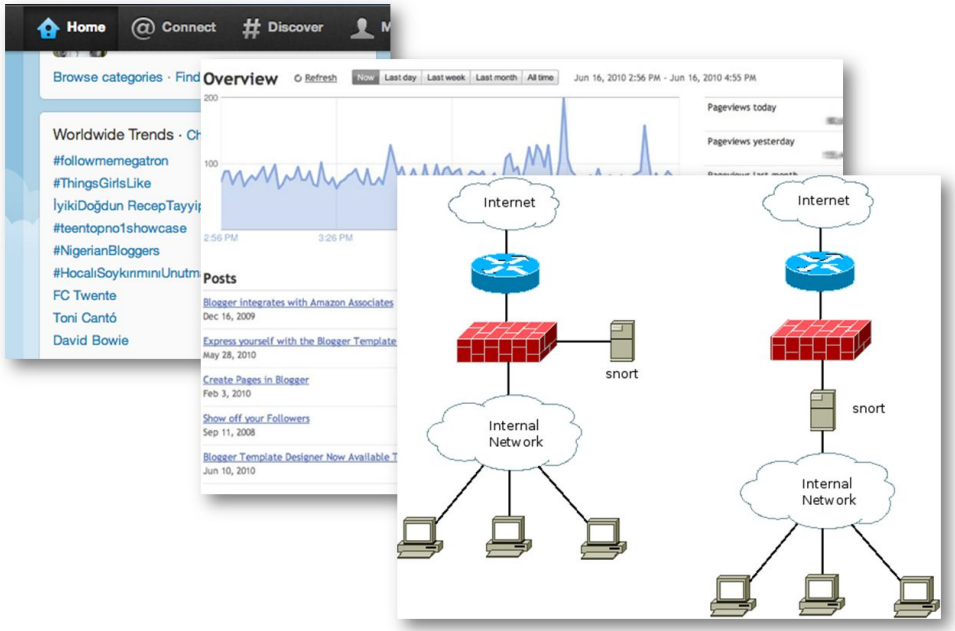
Spark Streaming

O que é Spark Streaming?

- Framework para processamento de streaming em larga escala
  - Suporta até 100 nós
  - Latência de segundos, em alguns casos
  - Integra-se com o modelo de processamento Spark em batch
  - integra-se também com KAFKA, Flume, ZeroMQ.

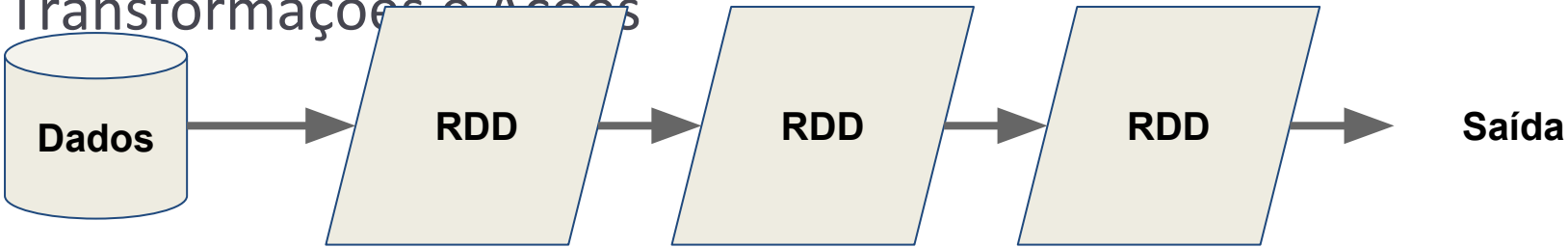
Motivação

- Análise de streaming de dados em tempo real.
  - Tendências em redes sociais
  - Estatísticas web
  - Detecção de intrusão de sistemas
  - etc.



Lembrando RDD

- Abstração fornecida pelo Spark para a manipulação de dados.
- Representação de um dado distribuído pelos nós do cluster que pode ser operado em paralelo.
- Transformações e Ações



RDD - Resilient Distributed Dataset

Particionado

Dividido em nós de um cluster

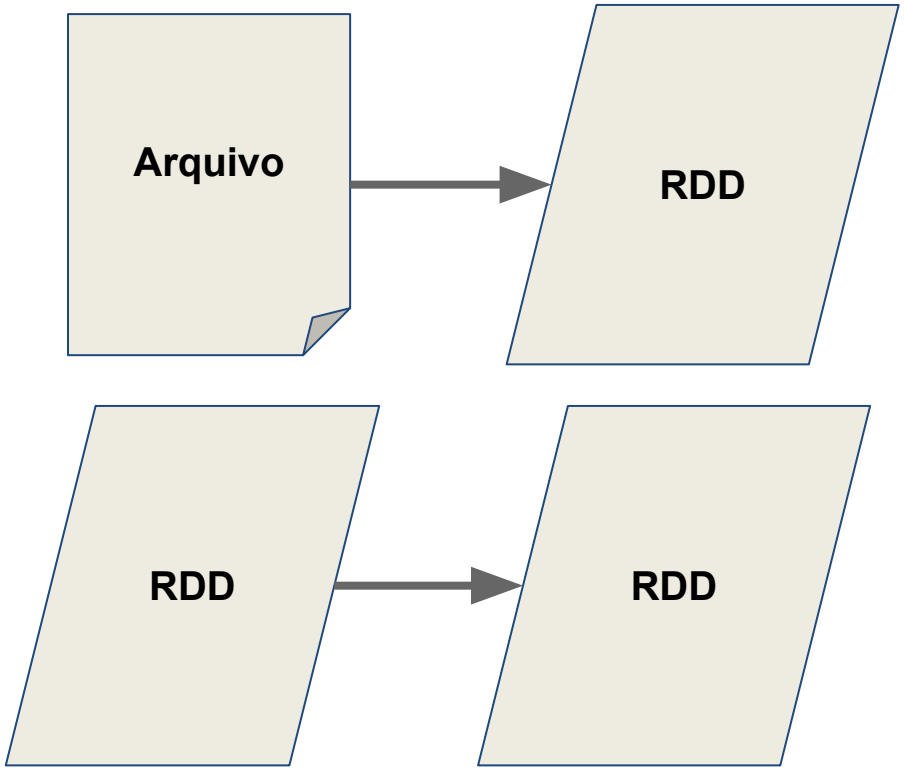
Imutável

RDDs, uma vez criados, não podem ser alterados

Resiliente

Podem ser reconstruídos mesmo se um nó caia

Criando um RDD




```
rdd = sc.textFile("data.txt")
```

```
rdd2 = rdd.distinct()
```

Streaming de Dados

```
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
```

Spark Streaming

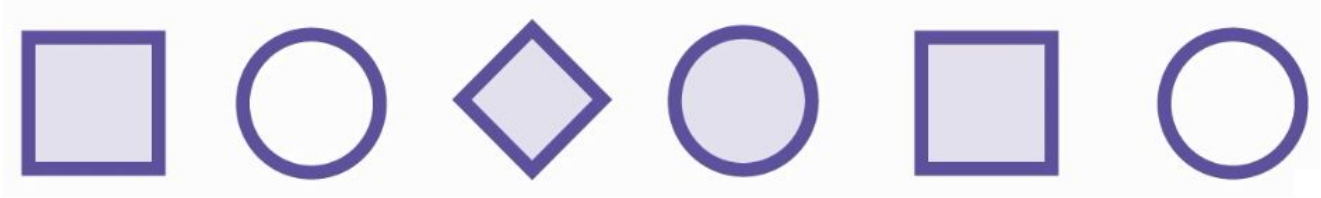
2016-12-30 09:09:58,239 INFO	org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServer: Listening HTTP traffic on /0.0.0.0:50075	HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:56745	2016-12-30 09:09:58,037 INFO org.mortbay.log: Started	2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26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

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

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

DStream - Discretized Stream

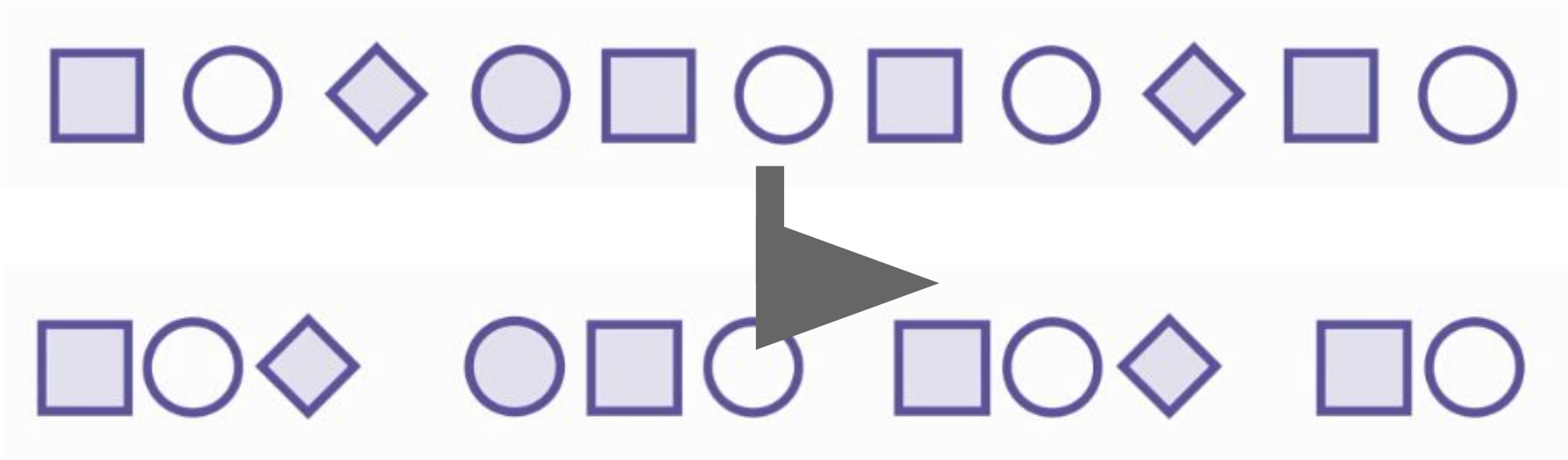
2016-12-30 09:09:58,239 INFO	org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServer: Listening HTTP traffic on /0.0.0.0:50075	HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:56745	2016-12-30 09:09:58,037 INFO org.mortbay.log: Started	2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26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



Stream “discretizado” = DStream = Sequência de RDDs

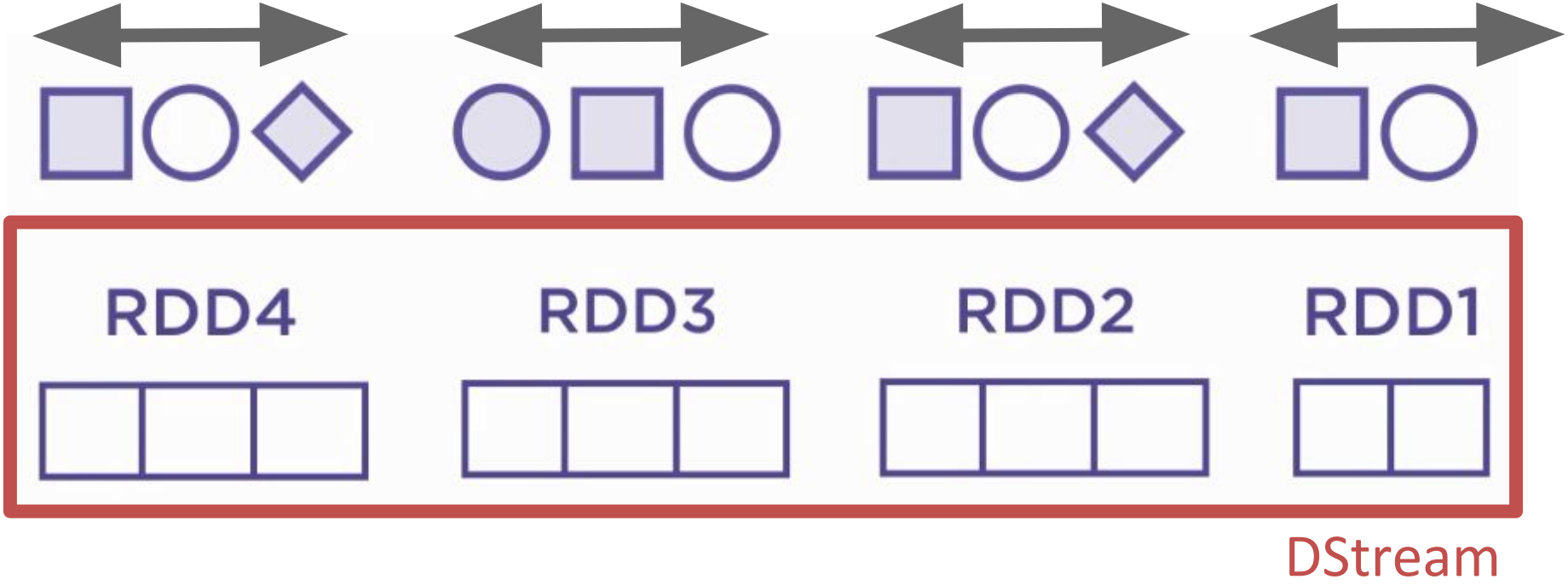
DStream - Discretized Stream

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



DStream - Discretized Stream

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

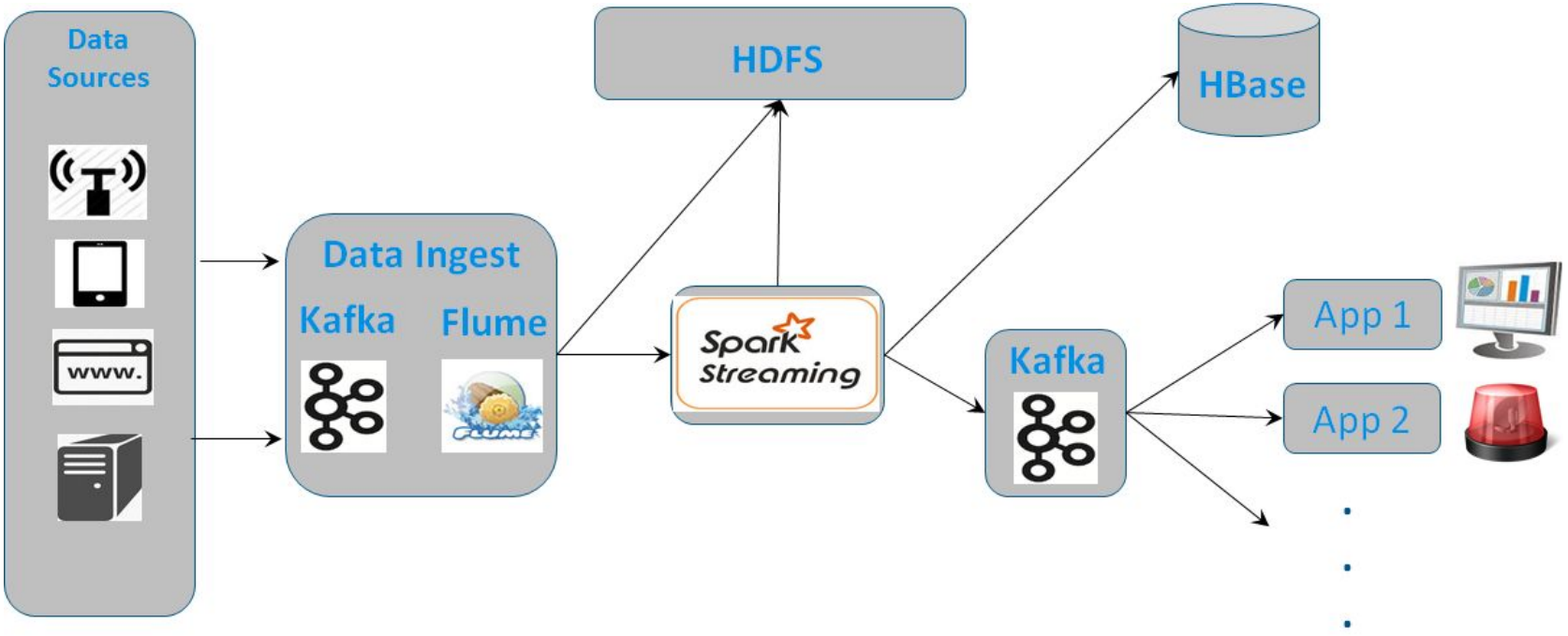


Stateless vs Stateful

Transformações são aplicadas em um único RDD

Transformações são acumuladas em múltiplos RDD

Arquitetura do Processamento de Streaming de Dados



Configurando Hadoop e Spark no Windows!!!

Pré Requisitos para Windows

- ★ HADOOP 3.0.1
  - http://hadoop.apache.org/releases.html
  - binary 3.0.1
  - http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.0.1/hadoop-3.0.1.tar.gz
  - Abrir hadoop-3.0.1.tar.gz
  - Abrir pasta hadoop-3.0.1
  - Extrair conteúdo desta pasta para **c:\bigdata\hadoop**
- ★ WINUTILS para Haddop 3.0.0 ou superior
  - https://github.com/steveloughran/winutils/tree/master/hadoop-3.0.0
  - download **winutils.exe** para c:\bigdata\hadoop\bin

Pré Requisitos para Windows

- ★ SPARK 2.3.0
  - https://spark.apache.org/downloads.html
  - Pre-built for Apache Hadoop 2.7 and later
  - Direct Download
  - Abrir spark-2.3.0-bin-hadoop2.7.tgz
  - Abrir pasta spark-2.3.0-bin-hadoop2.7
  - Extrair conteúdo desta pasta para **c:\bigdata\spark**
- ★ JAVA JRE
  - java -version
    - Deve ser 7 ou 8 (9 ainda não funciona)

Setup

```
>set HADOOP_HOME=C:\bigdata\hadoop
>set SPARK_HOME=c:\bigdata\spark
>cd %HADOOP_HOME%\bin
>hdfs namenode -format
c:\bigdata\hadoop\bin>winutils ls c:\tmp\
drwxrwxrwx 1 LSBD\manoel.ribeiro LSBD\Domain Users 0 Oct
3 2017 c:\tmp\hive

c:\bigdata\hadoop\bin>winutils chmod 777 c:\tmp\
```

Iniciando

```
>cd %SPARK_HOME%\bin
c:\bigdata\spark\bin>pyspark
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Python version 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014
22:15:05)
SparkSession available as 'spark'.
>>>
```

Hello World!

```
>>> data=[(‘Iphone8’, 5000),(‘Pixel2’,
4000),(‘GalaxyS8’,3000), (‘MotoZ’, 2500)]

>>> df=spark.createDataFrame(data,(‘smartphone’,’valor’))
>>> df.printSchema()
root
 |-- smartphone: string (nullable = true)
 |-- valor: long (nullable = true)
```

Spark Streaming - Prática

Exemplo 1 (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
```

Aparece a string **ERROR**?

Exemplo 1 (filtragem)

Terminal 2

4. Instalar NetCat da pasta streaming
5. 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.

   - `ncat -lk 9999`
6. DIGITAR LIVRE
7. ESPORADICAMENTE A PALAVRA ERROR

Exemplo 1 (filtragem)

Terminal 1

5. Executar sparkStreaming.py
   - `cd \bigdata\spark\bin`
   - Baixar e examinar o código
     - https://raw.githubusercontent.com/antoniomralmeida/streaming/master/sparkStreaming.py
   - `spark-submit sparkStreaming.py localhost 9999`

Fim