FACULDADE iBRATEC

Ecossistema Hadoop

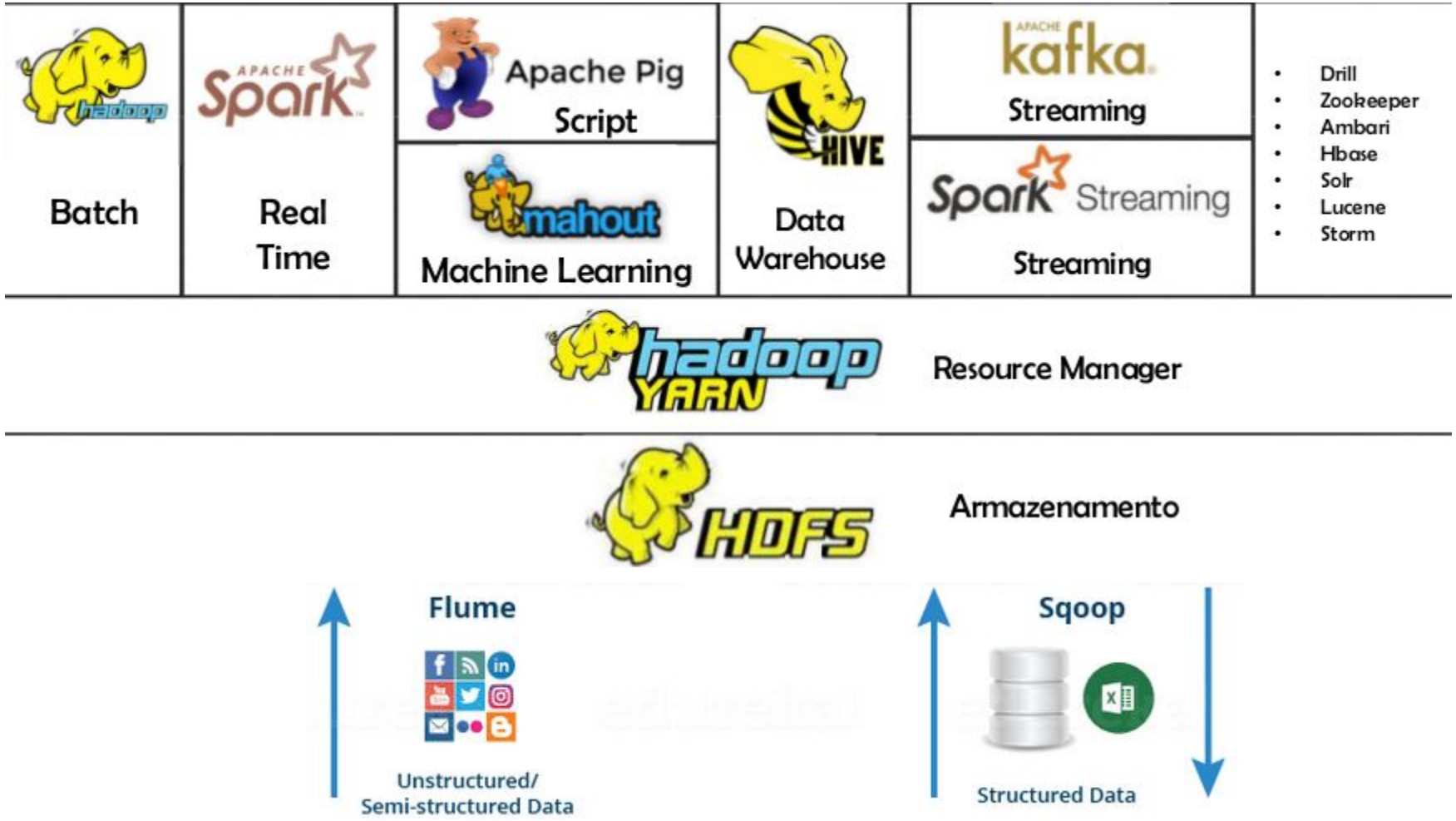
MSc. Wesley Lima

Roteiro

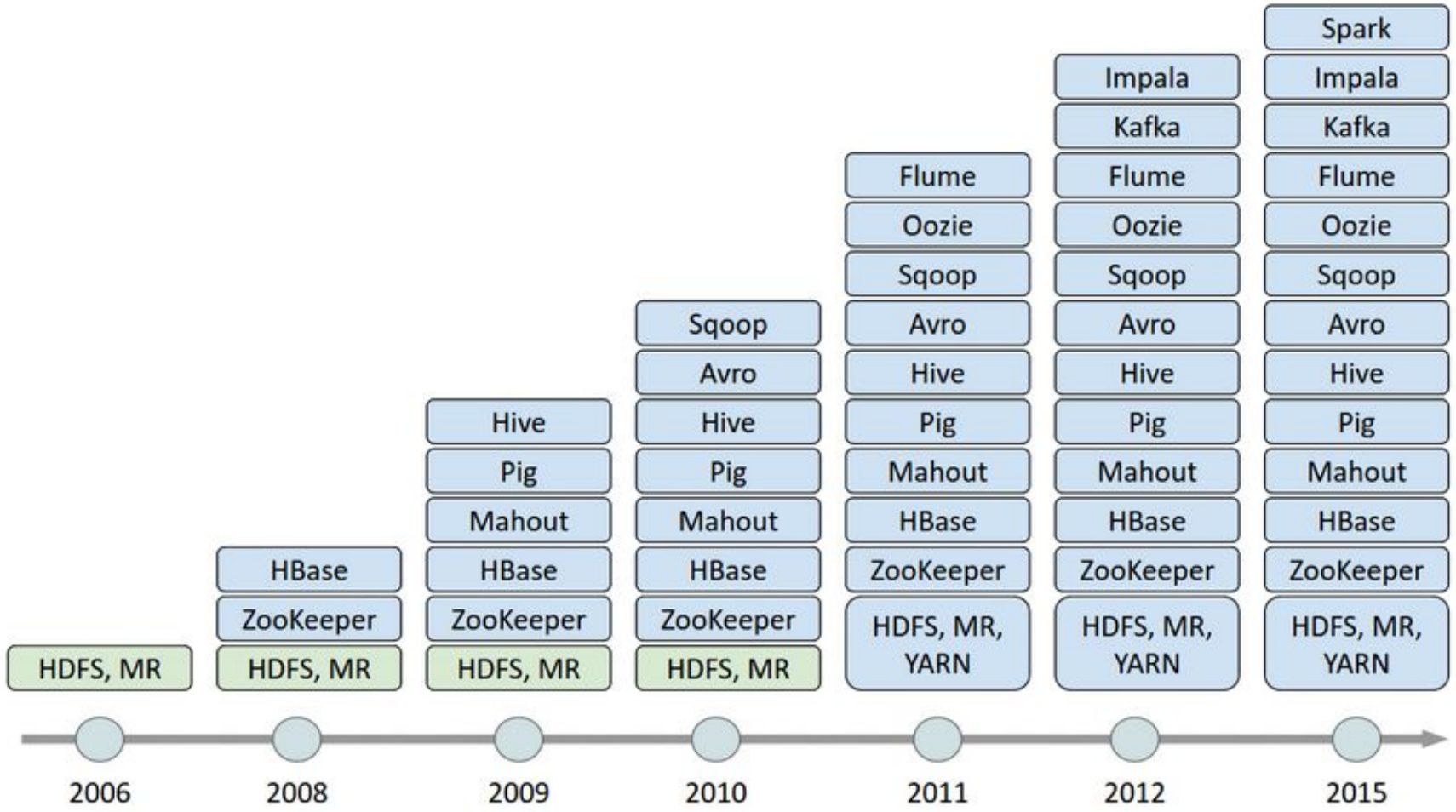
- Ecossistema Hadoop
- HDFS
- MapReduce
- Sqoop
- Hive
- Pig
- Flume
- Spark
- Spark streaming

FACULDADE iBRATEC

Ecossistema Hadoop

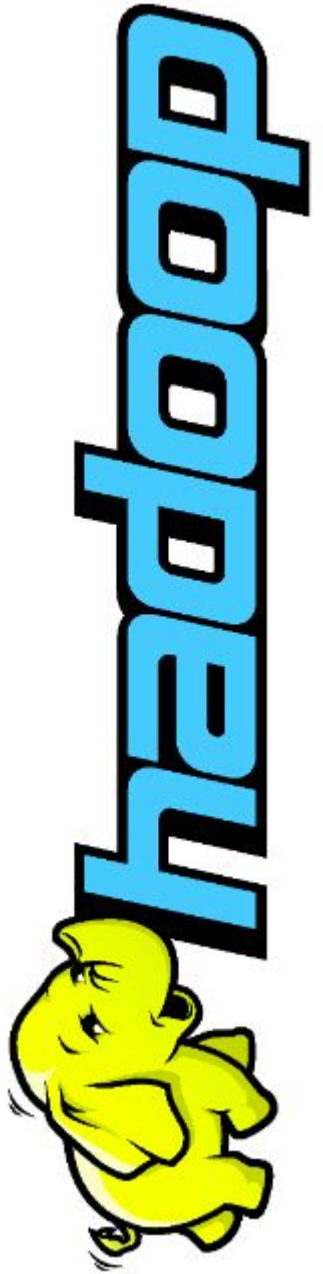


Ecossistema Hadoop



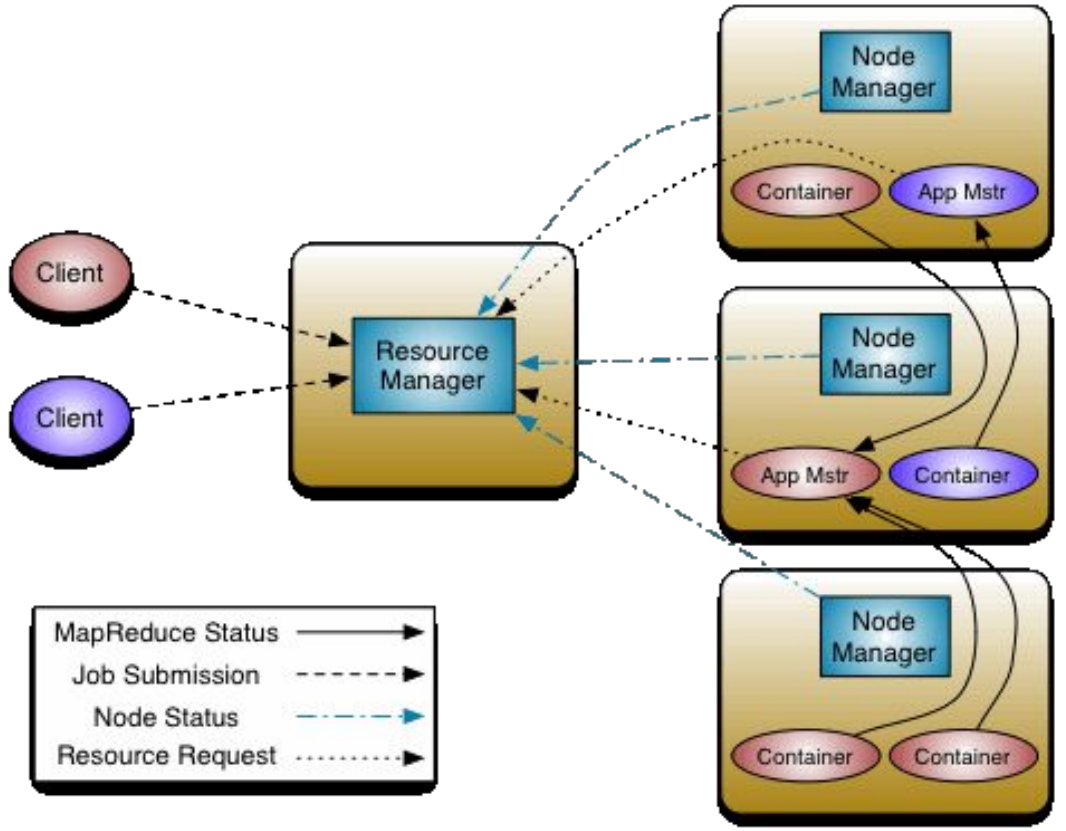
Hadoop

- **Desenvolvido em Java;**
- **Open source;**
- **Processamento em Batch;**
- **Baseado no Conceito de MapReduce;**
- **Capaz de distribuir o processamento em em dezenas ou milhares de nós em um cluster;**
- **Suporte a dados estruturados e não estruturados.**



Yarn

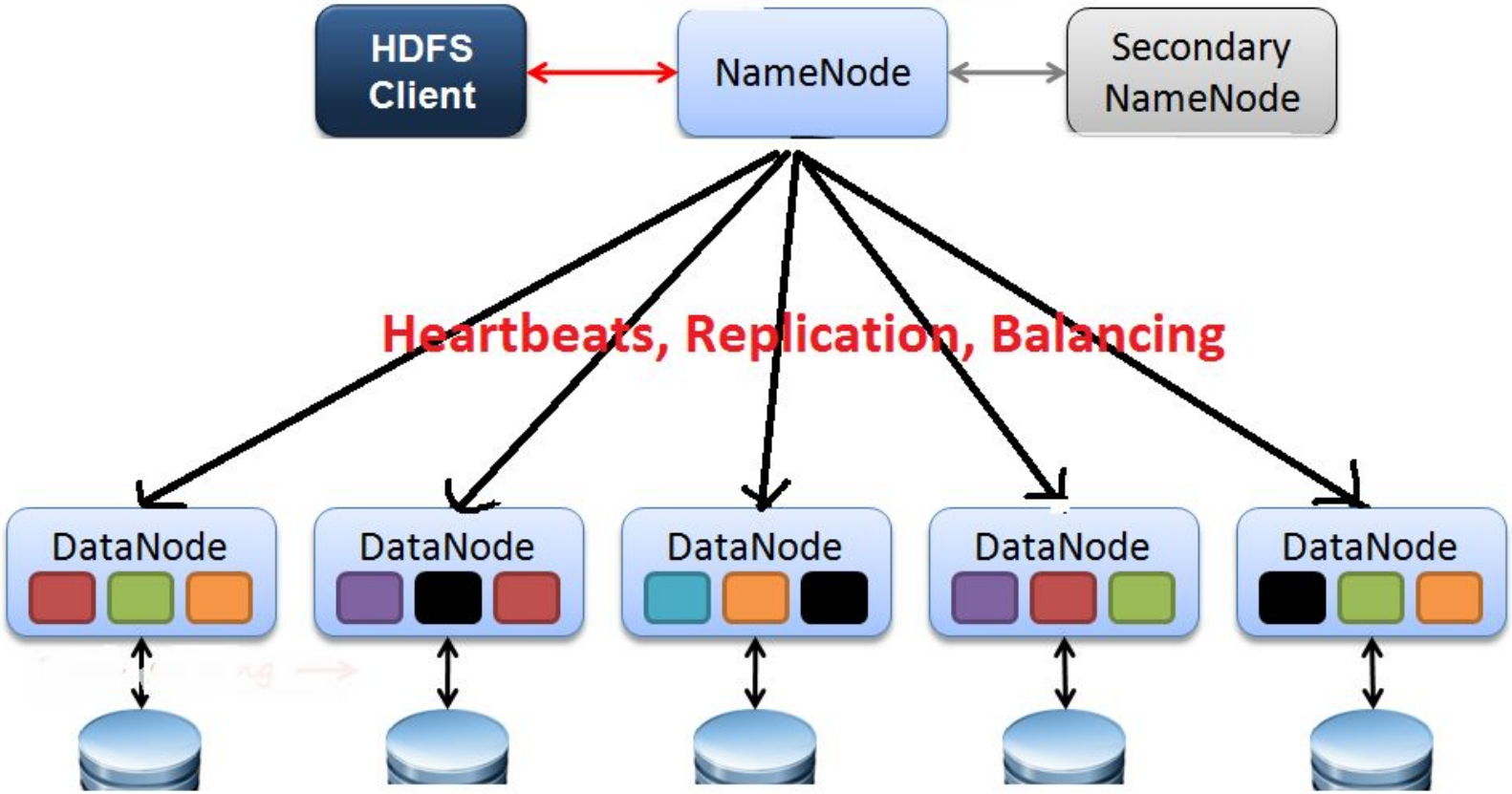
- **Alocação de recursos de forma global e unificada nos clusters;**
- **Agendamento;**
- **Priorização;**
- **Tolerância a falhas.**



HDFS

- **Sistema de arquivos distribuídos do hadoop;**
- **Armazena dados em blocos;**
- **Replicação transparente (default 3 nós);**
- **Acesso diferenciado (não há compatibilidade direta);**
- **Linhas de comando;**
- **Você pode copiar arquivos de um sistema para outro.**

HDFS


HDFS Client
NameNode
Secondary NameNode
Heartbeats, Replication, Balancing
DataNode
DataNode
DataNode
DataNode
DataNode

HDFS (principais comandos)

Comando	Descrição	Parâmetros	Exemplo
`-ls`	Lista conteúdo de diretório	-d listagem simples -R recursivo	`hdfs dfs -ls -d /`
`-put`	Copia arquivo do sistema local para o hdfs		`hdfs dfs -put /user/file.txt /cloudera/filte.txt`
`-mv`	Move arquivo ou diretório do sistema local para o hdfs		`hdfs dfs -put file.txt /cloudera`
`-rm`	Remove arquivo ou pasta	-r exclui de forma recursiva	`hdfs dfs -rm /cloudera/file.txt`

HDFS (principais comandos)

Comando	Descrição	Parâmetros	Exemplo
`-du`	Verifica tamanho do arquivo		`hdfs dfs -du input.txt`
`-cat`	Exibe conteúdo do arquivo		`hdfs dfs -cat /user/file.txt`
`-mkdir`	Cria uma pasta	-p cria caminho	`hdfs dfs -mkdir /cloudera/diretorio`
`-rmdir`	Remove diretório		`hdfs dfs -rmdir /cloudera/diretorio`
`-tail`	Mostra parte final de arquivo		`hdfs dfs -tail /cloudera/file.txt`

HDFS (principais comandos)

Comando	Descrição	Parâmetros	Exemplo
`-count`	Conta o número de diretórios, arquivos, etc.		`hdfs dfs -count /cloudera`
`-setrep`	Altera o fator de replicação de um arquivo		`hdfs dfs -setrep 3 /user/file.txt`
`-stat`	Informa estatísticas do arquivo os diretório	%F: tipo %n: nome %r: fator de replicação %y: data de modificação	`hdfs dfs -stat %F /cloudera/file.txt`
`-fsck`	Verifica o estado do sistema de arquivos		`hdfs dfs -fsck /`

HDFS (Como adicionar um arquivo?)

```
#criar diretorio no hdfs
[cloudera@quickstart ~]$ hdfs dfs -mkdir /contar/

#entrar no diretorio do arquivo
[cloudera@quickstart ~]$ cd Downloads

#adicionar o arquivo pesquisa.txt no hdfs
[cloudera@quickstart Downloads]$ hdfs dfs -put pesquisa.txt
/contar/pesquisa.txt

#verificar o que tem no diretorio contar
[cloudera@quickstart ~]$ hdfs dfs -ls /contar/
```

HDFS (Como adicionar um arquivo?)

```
#verificar fator de replicacao de pesquisa.txt
[cloudera@quickstart ~]$ hdfs dfs -stat %r /contar/pesquisa.txt

#alterar fator de replicacao
[cloudera@quickstart ~]$ hdfs dfs -setrep 3 /contar/pesquisa.txt

#ver o arquivo
[cloudera@quickstart ~]$ hdfs dfs -cat /contar/pesquisa.txt

#ver o arquivo
[cloudera@quickstart ~]$ hdfs dfs -tail /contar/pesquisa.txt
```

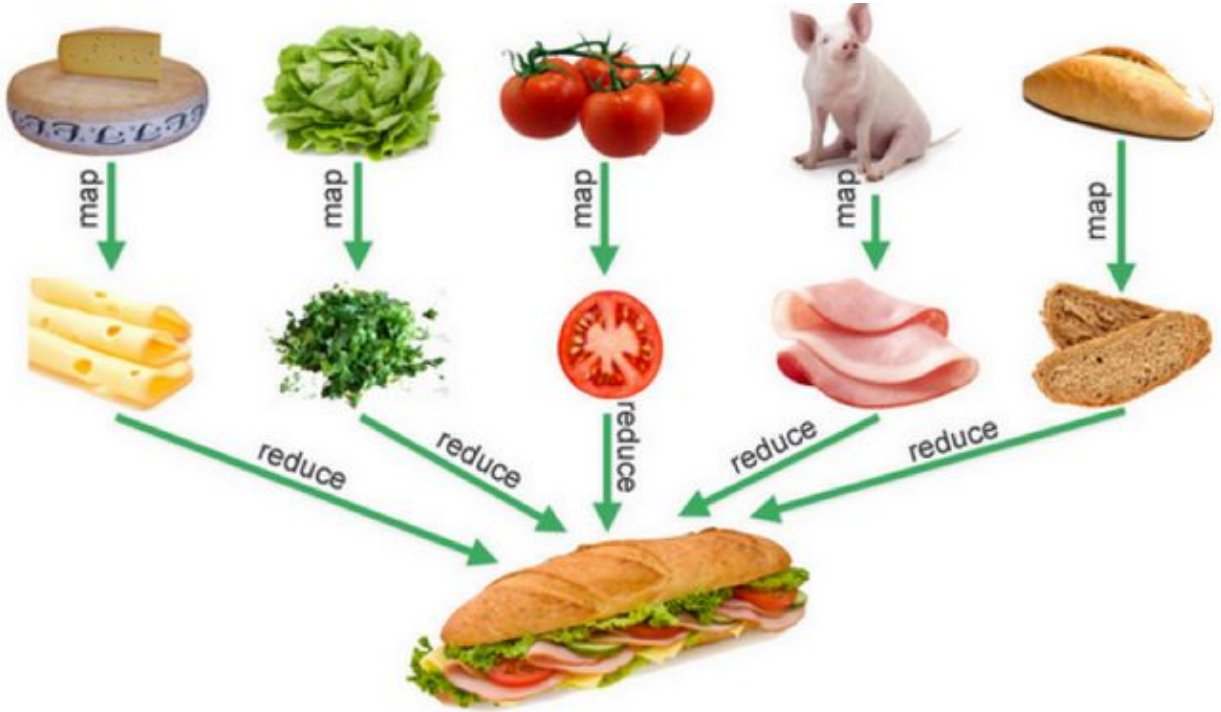
MapReduce

- **É um modelo de programação: permite que grandes volumes de dados sejam processados por meio da divisão de uma aplicação em tarefas independentes, executados em paralelo nos servidores do cluster;**

- **Dividir tarefas de processamento de dados em vários nós:**
  - **Dados são divididos em blocos;**
  - **Divisão de problemas grandes e/ou complexos em pequenas tarefas;**

MapReduce

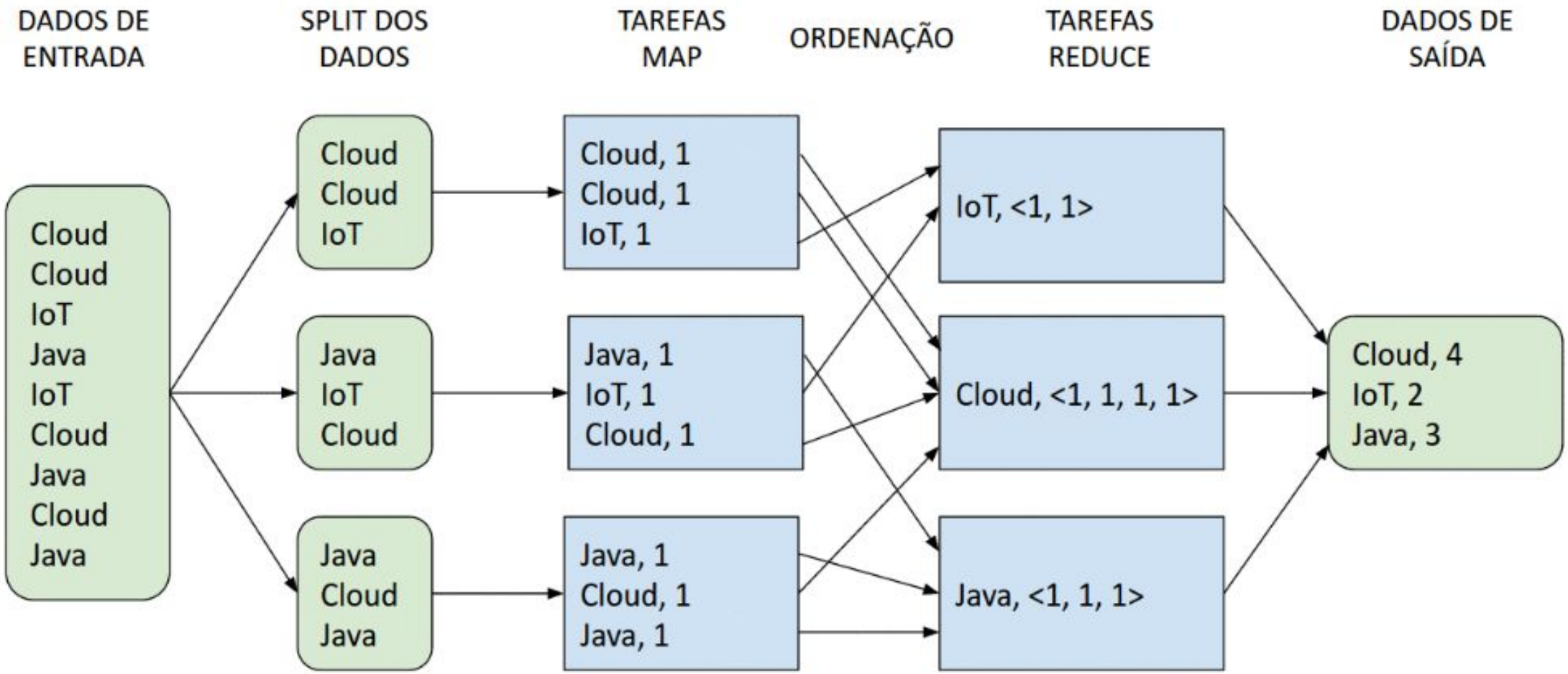
- **Escalável;**
- **Tolerante a falhas;**
- **Disponibilidade;**
- **Confiável;**
- **Usa conceito de chave valor;**
- **Não cria gargalos na rede, pois dados não trafegam (processamento em nó).**



MapReduce

- **Mapeamento é executado em paralelo nos nós;**
- **Apenas quando Mapeamento é encerrado, redução inicia, também em paralelo;**
- **Fase intermediária: Suffle ;**
- **Existem tarefas que requerem apenas a etapa de Mapeamento;**

MapReduce


DADOS DE ENTRADA
SPLIT DOS DADOS
TAREFAS MAP
ORDENAÇÃO
TAREFAS REDUCE
DADOS DE SAÍDA
Cloud
Cloud
IoT
Java
IoT
Cloud
Java
Cloud
Java
Cloud
Cloud
IoT
Java
IoT
Cloud
Java
Cloud
Java
Cloud, 1
Cloud, 1
IoT, 1
Java, 1
IoT, 1
Cloud, 1
Java, 1
Cloud, 1
Java, 1
IoT, <1, 1>
Cloud, <1, 1, 1, 1>
Java, <1, 1, 1>
Cloud, 4
IoT, 2
Java, 3

MapReduce (em python)

```
#! /usr/bin/python
#script para o mapper
import sys

#le cada linha de stdin
for line in sys.stdin:
    line = line.strip()
    #pega as palavras de cada linha
    words = line.split()
    #gera o contador para cada palavra
    for word in words:
        #escreve o par key-value para ser processado no reducer
        print'%s\t%s'%(word, 1)
```

MapReduce (em python)

```
#! /usr/bin/python
#script para o reducer
import sys

current_word = None
current_count = 0
word = None

#processa cada par key-value do mapper
for line in sys.stdin:
    line = line.strip()
     #obtem cada key e value da linha corrente
    word, count = line.split('\t', 1)
```

MapReduce (em python)

```
#continuacao script para o reducer
#processa cada par key-value do mapper
    try:
      count = int(count)
    except ValueError:
      continue
    if current_word == word:
      current_count += count
    else:
      if current_word:
        print "%s\t%s" % (current_word, current_count)
      current_count = count
      current_word = word
```

MapReduce (em python)

```
#continuacao script para o reducer

#output para o count da ultima palavra

if current_word == word:
   print "%s\t%s" % (current_word, current_count)



#salve os arquivos como mapper.py e reducer.py

#antes de executar o codigo e preciso tornar os scripts
#executaveis
```

MapReduce (em python)

```
#executar no shell

#tornar script python executavel
[cloudera@quickstart ~]$ chmod +x mapper.py
[cloudera@quickstart ~]$ chmod +x reducer.py

#testando localmente
[cloudera@quickstart ~]$ echo 'jack be minble jack be
quick'|python  mapper.py | sort -t 1 | python reducer.py
```

MapReduce (em python)

```
#executar no shell
#testando com o hadoop
[cloudera@quickstart ~]$ hadoop jar
/usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.13.0.ja
r -file /home/cloudera/mapper.py -mapper mapper.py -file
/home/cloudera/reducer.py -reducer reducer.py -input
/contar/pesquisa.txt -output /contar3

#verificando o que foi salvo no hdfs
[cloudera@quickstart ~]$ hdfs dfs -ls /contar3

#abrindo o resultado
[cloudera@quickstart ~]$ hdfs dfs -cat /contar3/part-00000
```

MapReduce (em python com MRJob)

```
#preparar o ambiente - Software Collection
#para instalar o python3.6 seguir as instrucoes

#https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-local-programming-environment-on-centos-7

#no passo - ATENCAO:
[cloudera@quickstart ~]$ sudo yum -y install
https://centos6.iuscommunity.org/ius-release.rpm #Baixar-site

#no passo - Refazer:
[cloudera@quickstart ~]$sudo yum -y install ius-release-el6.rpm
```

MapReduce (em python com MRjob)

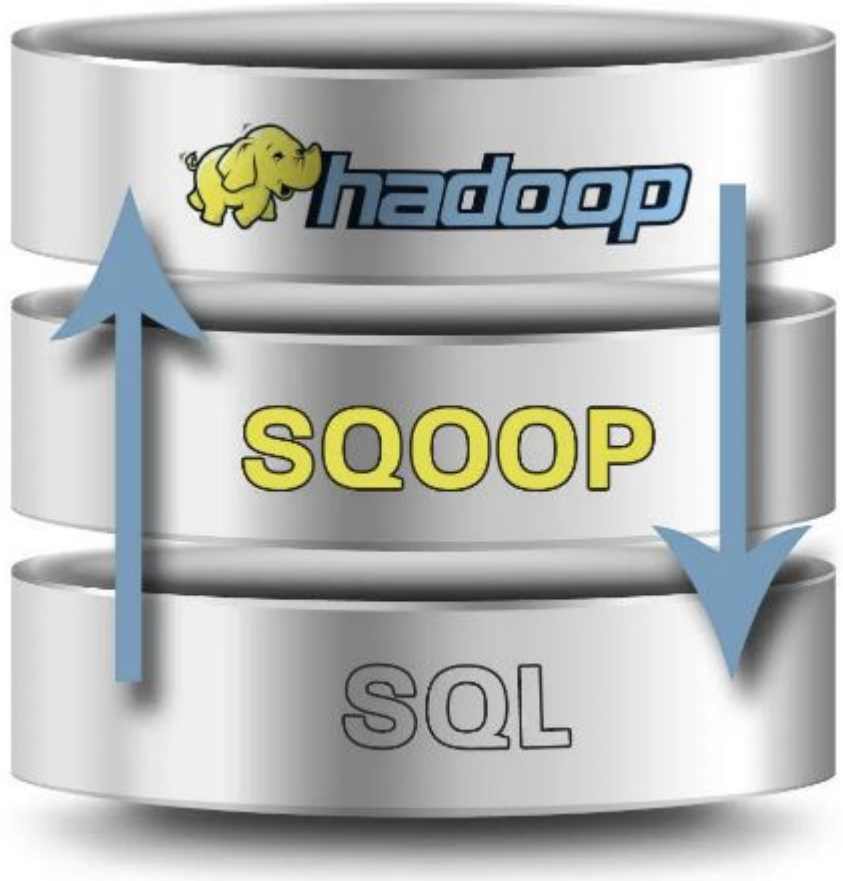
```
#script
#! /usr/bin/python3.6
from mrjob.job import MRJob

class MRWordCount(MRJob):

    def mapper(self, _, line):
      for word in line.split():
         yield(word,1)
    def reducer(self, word, counts):
      yield(word, sum(counts))
if __name__ == '__main__':
    MRWordCount.run()
```

MapReduce (em python com MRjob)

```
#preparar o ambiente - Software Collection

#tornar executavel:
[cloudera@quickstart ~]$ chmod +x word_count.py

#testar localmente:
[cloudera@quickstart ~]$  python3.6 word_count.py pesquisa.txt
```

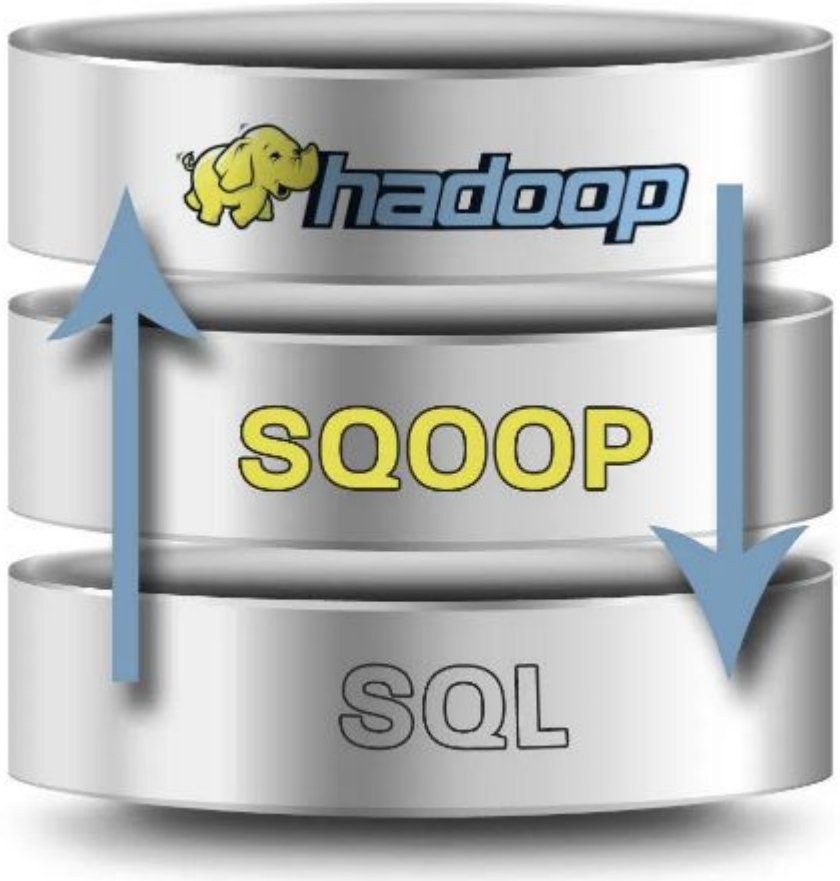
Sqoop

- **Pertence ao ecossistema hadoop;**
- **Usado quando se pretende importar dados para:**
  - **Hive;**
  - **Hbase;**
  - **Accumulo;**
  - **HDFS.**



Sqoop

- **Importação de Dados de RDBMS;**
- **Utiliza um drive JDBC;**
- **A importação é baseada em tabelas;**
- **Pode ter um filtro;**
- **Armazena em arquivo texto delimitado (virgula), binário (Arvro), ORC, et.**



Sqoop

```
#configurando o ambiente

#logar no postgres
[cloudera@quickstart ~]$ sudo -u postgres psql
#mudar a senha de usuario postgres
postgres=# alter user postgres PASSWORD '123456';
postgres=# \q

#alterar arquivo de configuração do postgres
[cloudera@quickstart ~]$ sudo vi
/var/lib/pgsql/11/data/pg_hba.conf
#output
# IPv4 local connections:      #inserir essa lina
host all          all          0.0.0.0/0               md5
host all          all          127.0.0.1/32            ident
```

Sqoop

```
#configurando o ambiente

#reiniciando o postgres
[cloudera@quickstart ~]$ sudo service postgresql-11 stop
[cloudera@quickstart ~]$ sudo service postgresql-11 start

#Baixar o drive JDBC-usado pelo sqoop para acessar o postgres
[cloudera@quickstart Downloads]$ wget
http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.j
ar

#copiar o drive para a pasta /var/lib/sqoop
[cloudera@quickstart Downloads]$ sudo cp
postgresql-9.2-1002.jdbc4.jar
/var/lib/sqoop/postgresql-9.2-1002.jdbc4.jar
```

Sqoop

```
#iniciando no sqoop

#listar base de dados com sqoop
[cloudera@quickstart ~]$ sqoop list-databases --connect
jdbc:postgresql://127.0.0.1/ --username postgres -password
123456

#listar tabelas de uma base de dados com sqoop
[cloudera@quickstart ~]$ sqoop list-tables --connect
jdbc:postgresql://127.0.0.1/bigdata --username postgres
-password 123456 -- --schema dimensional

#importar banco de dados do postgres para o HDFS com sqoop
[cloudera@quickstart ~]$ sqoop import-all-tables --connect
jdbc:postgresql://127.0.0.1/bigdata --username postgres
-password 123456 -- --schema dimensional
```

Sqoop

```
#iniciando no sqoop

#visualizar no HDFS
[cloudera@quickstart ~]$ sudo hdfs dfs -ls /user/cloudera

#visualizando o diretorio fatovendas
[cloudera@quickstart ~]$ sudo hdfs dfs -ls
/user/cloudera/fatovendas/

#usando cat para ve um arquivo
[cloudera@quickstart ~]$ sudo hdfs dfs -cat
/user/cloudera/fatovendas/part-m-00000
```

Hive

- **Data Warehouse com hadoop;**
- **Suporta grande volume de dados: distribuído;**
- **Gera tarefa map reduce para maioria das consultas;**
- **HiveSQL: linguagem de consulta parecido com SQL;**
- **Pode-se utilizar:**
  - **Linha de comando;**
  - **Web (Ambar ou Hue).**

Hive

- Dados estruturados;
- Feito para inserção e leitura (não atualização);
- Gera tarefa map reduce para maioria das consultas;
- Arquitetura;
  - Banco de dados;
  - Tabelas;
  - Partição: equivalente a coluna, podendo dividir dados horizontalmente;
  - Bucket: partição vertical dos dados baseado em hashs;
- Fisicamente no HDFS: /user/hive/warehouse/

Hive

```
#logar no ambiente
[cloudera@quickstart ~]$ beeline
beeline>
beeline> !connect jdbc:hive2://
scan complete in 2ms
Connecting to jdbc:hive2://
Enter username for jdbc:hive2://: admin
Enter password for jdbc:hive2://: *****

#logado no shell do hive
0: jdbc:hive2://>
```

Hive

```
#alguns comandos
0: jdbc:hive2://> show databases;

#criar um banco de dados
0: jdbc:hive2://> create database bigdata;

#usar o banco de dados
0: jdbc:hive2://> use bigdata;

#hive possui schema: fazer a importacao ja na criacao do schema
0: jdbc:hive2://> create table dimensaocliente(chavecliente
int, idcliente int, cliente string, estado char(2), sexo
char(1), status string, datainiciovalidade date,
datafimvalidade date) row format delimited fields terminated by
‘,’ location ‘/user/cloudera/dimensaocliente’;
```

Hive

```
#consultar os primeiros dados de cliente
0: jdbc:hive2://> select * from dimensaocliente limit 10;

#criar a dimensao tempo
0: jdbc:hive2://> create table dimensaotempo(chavetempo int,
data date, dia int, mes int, ano int, diasemana int, trimestre
int) row format delimited fields terminated by ‘,’ location
‘/user/cloudera/dimensaotempo/’;

#criar a dimensao tempo
0: jdbc:hive2://> create table dimensaoproduto(chaveproduto
int, idproduto int, produto string, datainiciovalidade date,
datafimvalidade date) row format delimited fields terminated by
‘,’ location ‘/user/cloudera/dimensaoproduto/’;
```

Hive

```
#deletar uma tabela
0: jdbc:hive2://> drop table dimensaoproduto;

#criar a tabela vendedor
0: jdbc:hive2://> create table dimensaovendedor(chavevendedor
int, idvendedor int, nome string, datainiciovalidade date,
datafimvalidade date) row format delimited fields terminated by
‘,’ location ‘/user/cloudera/dimensaovendedor/’;

#criar a tabela fato
0: jdbc:hive2://> create table fatovendas(chavevendas int,
chavevendedor int, chaveclient int, chaveproduto int,
chavetempo int, quantidade int, valorunitario float, valortotal
float, desconto float) row format delimited fields terminated
by ‘,’ location ‘/user/cloudera/fatovendas/’;
```

Hive

```
#vamos explorar
0: jdbc:hive2://> show tables;

#visualizar o banco de dados diretorio hive
[cloudera@quickstart ~]$ hdfs dfs -ls /user/hive/warehouse/

#Usar a interface Web Hue
user: cloudera -- password: cloudera
```

Pig

- **Mais uma solução a complexidade do Hadoop;**
  - **Facilita o uso de mapreduce sem conhecer java;**
- **Linguagem de alto nível: Pig Latin para processamento de dados;**
- **Operação em memória;**
- **Forma de execução:**
  - **Interativa (shell - grunt);**
  - **Batch: script com extenção pig;**
  - **Embedded: dentro de outras aplicações**

Pig

- **Tipos de dados:**

Tipo	Descrição
int	Inteiro de 32 bits
long	Inteiro de 64 bits. L
biginteger	java big integer
bigdecimal	java decimal
float	Ponto flutuante de 3 bits. F
double	Ponto flutuante de 54 bits
boolean	Lógico
datetime	Data e Hora
chararray	String
bytearray	blob
Tuple	campos (fernando, 30)
Bag	Coleção de campos {(José,20),(Maria,12)}
Map	Conjunto de Chaves Valor ["nome"#"José','idade'#30]

Pig

```
#formas de rodar o pig
#x: roda comandos no shell
#x: roda comandos de um script
[cloudera@quickstart ~]$ pig -x[f] local
[cloudera@quickstart ~]$ pig -x[f] mapreduce

#iniciar o pig #abre o grunt
[cloudera@quickstart ~]$ pig -x mapreduce

#abrir dois arquivos de dados que estao no HDFS
grunt> dimensaocliente = LOAD '/user/cloudera/dimensaocliente/'
USING PigStorage(',') as (chavecliente:int, idcliente:int,
cliente:chararray, estado:chararray, sexo:chararray,
status:chararray, datainiciovalidade:datetime,
datafimvalidade:datetime);
```

Pig

```
#criar o objeto fatovendas
grunt> fatovendas = LOAD '/user/cloudera/fatovendas/' USING
PigStorage(',') as (chavevendas:int, chavevendedor:int,
chavecliente:int, chaveproduto:int, chavetempo:int,
quantidade:int, valorunitario:float, valortotal:float,
desconto:float);

#consultar
grunt> dump dimensaocliente;
grunt> dim10 = LIMIT dimensaocliente 10;
grunt> dump dim10;
grunt> describe dimensaocliente;
grunt> clienteporstatus = GROUP dimensaocliente by status;
grunt> dump clienteporstatus;
```

Pig

```
#algumas transformacoes nos dados
grunt> juncaocliente = JOIN fatovendas BY chavecliente,
dimensaocliente BY chavecliente;
grunt> dump juncaocliente;

#filtro
grunt> dimesaoclientePlatinum = FILTER dimensaocliente BY
status == 'Platinum';
grunt> dump dimensaoclientePlatinum;
grunt> clienteporstatus = GROUP dimensaocliente by status;
grunt> dump clienteporstatus;
```

Pig

```
#dividir
grunt> SPLIT dimensaocliente into dimensaoGold if status  ==
'Gold', dimensaoPlatinum if status == 'Platinum';
grunt> dump dimensaoGold;

#order
grunt>dimensaoclienteorder=ORDER dimensaocliente BY cliente ASC;
grunt> dump dimensaoclienteorder;

#salvar os dados em disco HDFS
grunt> STORE dimensaoclientePlatinum INTO '/user/cloudera/pig/'
USING PigStorage(',');
```

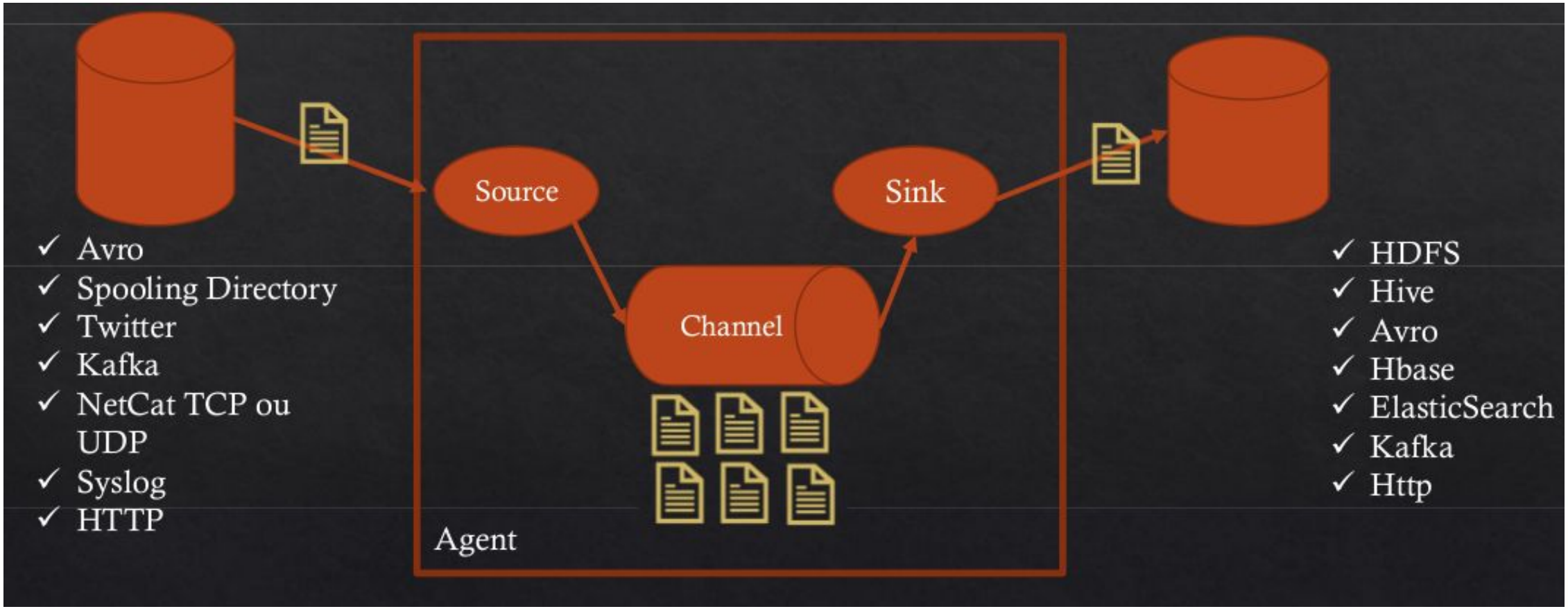
Flume

- **Ingestão de dados não estruturados;**
  - **Distribuído;**
  - **Confiável;**
  - **Alta disponibilidade;**
  - **Recuperação de falhas.**
- **Suporte de múltiplos tipos e destinos;**
- **Permite a construção de fluxo de dados;**
- **Gerencia de gargalos.**

Flume

- Event: unidade de dados
- Agent: processo que comporta o fluxo dos dados
- Source: Consome eventos: dados gerados por fonte externa
- Channel: Mantem uma fila de eventos até eles serem consumidos pelo Sink
- Sink: Remove o evento do Channel e o coloca no seu destino final

Flume


Source
Sink
Channel
Agent
✓ Avro
✓ Spooling Directory
✓ Twitter
✓ Kafka
✓ NetCat TCP ou UDP
✓ Syslog
✓ HTTP
✓ HDFS
✓ Hive
✓ Avro
✓ Hbase
✓ ElasticSearch
✓ Kafka
✓ Http

Flume

```
#iniciando com o flame
#criar dois diretorios um para o linux outro para o HDFS
[cloudera@quickstart ~]$ mkdir /home/cloudera/flume
[cloudera@quickstart ~]$ hdfs dfs -mkdir /user/flume

#criar arquivo de configuracao example.conf- salvar em cloudera
#example.conf esta em MaterialAula - executar o agente do flume
[cloudera@quickstart ~]$ sudo flume-ng agent --conf conf
--conf-file /home/cloudera/example.conf --name a1
-Dflume.root.logger=INFO,console

#output: ultima linha - ele esta monitorando a pasta -nao feche
[cloudera@quickstart ~]$ INFO
instrumentation.MonitoredCounterGroup: Component type: SINK,
name: k1 started
```

Flume

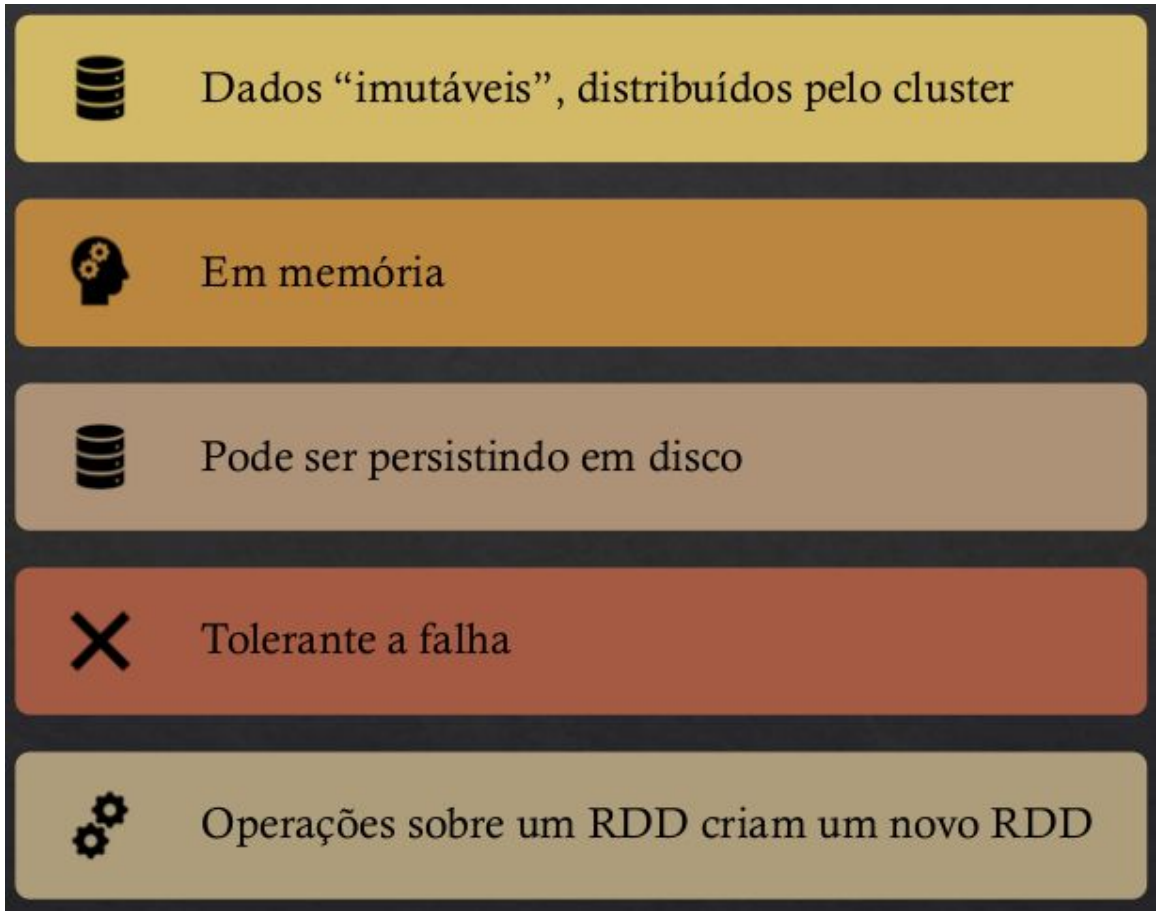
```
#iniciando com o flame
#criar um arquivo teste e salvar na pasta flame

#depois verifique o que aconteceu no HDFS - abra outro shell
[cloudera@quickstart ~]$ hdfs dfs -ls /user/flume/

#verifica o que tem no arquivo
[cloudera@quickstart ~]$ hdfs dfs -cat
/user/flume/FlumeData.1570249078364
```

Spark - Processamento em tempo real

- **Em tempo real/Interativo;**
- **Compatível com MapReduce;**
- **Compatível com Hadoop, no entanto não é pré-requisito;**
- **Em memória / disco / híbrido;**
- **Em cluster;**
- **Open Source;**
- **Shell em Scala ou Python;**
- **Estrutura de dados: RDD - Resilient Distributed Dataset**

Spark - Processamento em tempo real

- **Estrutura de dados: RDD - Resilient Distributed Dataset**



Spark - Processamento em tempo real

- **Principais componentes:**
  - **Core: Gestão de recursos, tolerância a falhas, monitoramento, agendamento;**
  - **SQL: Consulta por linguagem SQL para dados estruturados. Suporte a Hive;**
  - **Streaming: Processamento de dados em tempo real;**
  - **Mllib: Biblioteca de Machine Learning;**
  - **GraphX: Processamento de Grafos.**

Spark - Processamento em tempo real

- **Linguagens:**

Spark - Processamento em tempo real

- **Alguns comandos basicos:**

Transformação	Descrição
`filter()`	Aplica uma filtro
`map()`	Aplica uma função
`sample()`	Gera subconjunto
`distinct()`	Retorna elementos úncios
`intersection()`	Retornar interseção de dois RDDs
`subtract()`	Subtrai conteúdo de um RDD
`cartesian()`	Gera o produto cartesiano de 2 RDDs
`union()`	Gera RDD com elementos de 2 RDDs

Transformação	Descrição
keys()	Retorna apenas as chaves
values()	Retorna apenas os valores
groupByKey()	Agrupa por chaves
sortByKey()	Ordena por chave

Spark - Processamento em tempo real

- **Alguns comandos básicos:**

Ação	Descrição
collect()	Retorna todo o RDD
count()	Conta o número de elementos
countByValue()	Contagem agrupada
take()	Retorna o número de elementos passados como parâmetro
top()	Retorna os primeiros elementos de acordo com o parâmetro
reduce()	Combina elementos usando computação paralela
`mean()`	Calcula a média
`sum()`	Calcula a soma
`min()`	Menor Valor
`max()`	Maior Valor
`variance()`	Calcula a Variância

Spark - Processamento em tempo real

```
#iniciando com o spark
#abrir o shell do pyspark
[cloudera@quickstart ~]$ pyspark

#a variavel 'sc' de contexto: faz a conexao do shell com o
#cluster do spark
>>> sc
#criar  um rdd usando a variavel de contexto e a funcao
#paralellize
>>> numeros = sc.parallelize([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
>>> numeros
#acoes nao geram um novo RDD apenas transformacoes
>>> numeros.first()
```

Spark - Processamento em tempo real

```
#iniciando com o spark
#alguns comandos
>>> numeros.take(5)
>>> numeros.top(5)
>>> numeros.collect()

#algumas operacoes
>>> numeros.count()
>>> numeros.mean()
>>> numeros.sum()
>>> numeros.max()
>>> numeros.min()
>>> numeros.stdev()
```

Spark - Processamento em tempo real

```
#iniciando com o spark
#forcando a persistencia dos dados
#Existe diferentes formas de persistencia - consulte
#vamos persistir em memoria de forma serializada
>>> numeros.persist(StorageLevel.MEMORY_ONLY_SER)

#retirar a persistencia dos dados
>>> numeros.unpersist()

#transformacoes
>>> filtro = numeros.filter(lambda filtro: filtro > 2)
>>> filtro.collect()
```

Spark - Processamento em tempo real

```
#iniciando com o spark - executar o MapReduce com spark
#pegar o arquivo pesquisa.txt do diretorio Downloads
>>> pesquisa =
sc.textFile("file:///home/cloudera/Downloads/pesquisa.txt")
>>> pesquisa.take(10)

#fazer o mapReduce em 3 etapas
>>> contagem = pesquisa.flatMap(lambda palavra: palavra.split("
"))
>>> contagem = contagem.map(lambda pal: (pal,1))
>>> contagem = contagem.reduceByKey(lambda a, b: a + b)
>>> contagem.take(5)
```

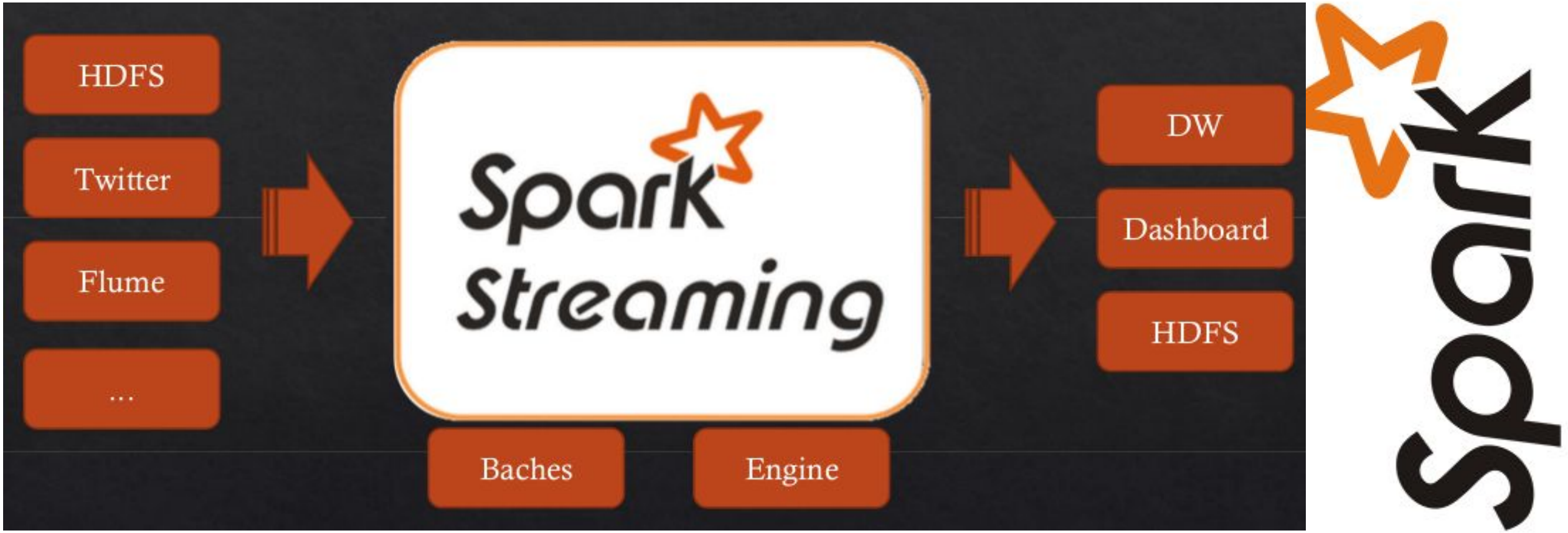
Spark - Processamento em tempo real

```
#iniciando com o spark - executar o MapReduce com spark

#vamos persistir esse resultado
>>> contagem.saveAsTextFile("conta")
>>> quit()

#ve a saida no HDFS
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/conta
[cloudera@quickstart ~]$ hdfs dfs -cat
/user/cloudera/conta/part-00000
```

Spark - Processamento em tempo real

- **O que é Streaming;**
  - **Dados processados de forma contínua;**
  - **Em tempo real, ou próximo ao tempo real;**
  - **Exemplos:**
    - **Dados de sensores;**
    - **Logs de acesso (detecção de invasão);**
    - **Transações financeiras (busca de fraudes).**

Spark - Processamento em tempo real



Spark - Processamento em tempo real

```
#iniciando com o spark streaming - mapreduce
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
sc = SparkContext("local[2]", "Contagem")
ssc = StreamingContext(sc,10)
pesquisa = ssc.textFileStream("file:///home/cloudera/spark/")
contagem = pesquisa.flatMap(lambda palavra: palavra.split(" "))
contagem = pesquisa.map(lambda pal: (pal,1))
contagem = contagem.reduceByKey(lambda a,b: a + b)
contagem.pprint()
ssc.start()
ssc.awaitTermination()
```

Spark - Processamento em tempo real

```
#iniciando com o spark streaming
#criar um diretorio spark em cloudera
[cloudera@quickstart ~]$ mkdir spark

#chamar o spark no shell via script python
[cloudera@quickstart ~]$ spark-submit
/home/cloudera/exemplo_streaming.py

#salvar alguns arquivos na pasta spark e ve o que acontece
```

Questões

1. **Fale sobre o hadoop e as vantagens de empregar as ferramentas do seu ecossistema.**
2. **O que é MapReduce?**
3. **Explique o HDFS?**
4. **O que é Sqoop e Flume?**
5. **Fale sobre o Hive?**
6. **Explique quando é empregado o Pig?**
7. **Descreva os principais usos do Spark?**

Obrigado

Até a próxima aula...

FACULDADE iBRATEC