Sumário

[Módulo 1 - Introdução 1](#_Toc78476950)

[Módulo 2 – Modelos Relacionais 4](#_Toc78476951)

[Módulo 3 – Modelos Dimensionais 7](#_Toc78476952)

[Módulo 4 – NoSQL 9](#_Toc78476953)

[MongoDB. 9](#_Toc78476954)

[REDIS 13](#_Toc78476955)

[MÓDULO 7 – Processamento em Batch com Hadoop 19](#_Toc78476956)

[Módulo 8 – SQOOP, HIVE e PING 25](#_Toc78476957)

[MÓDULO 9 – PING e FLUME 29](#_Toc78476958)

[Módulo 10 – Processamento Real Time e Streaming com Spark. 32](#_Toc78476959)

Engenharia de Dados

# Módulo 1 - Introdução

Disponibilidade dos Dados

Confiabilidade dos Dados

Segurança dos Dados

Planejamento dos Dados

Monitorar a performance

Desenvolver Data Lake e DataWareHouse

O que é Big Data?

Dados produzidos pelos 5V – Velocidade, Volume , Variedade, Veracidade e Valor.

Estrutura dos Dados

Estruturados => Os dados possuem um padrão.

Semi Estruturados => Mistura heterogênea, porém não é rígida. Ex: Arquivo XML.

Não Estruturado => Não existe restrição dos dados. Ex: Vídeos, Twitters.

Arquitetura dos Dados

Ambiente Transacional => Registrar Fatos -> Recuperar Registros.

Ambiente Analítico => Manter Histórico -> Analisar Dados.

Cluster

Computação distribuídas. Vários computadores(nós) executando simultaneamente.

Podem estar dispersos fisicamente em países.

Estrutura Master/Slace => Um coordena e outra processa.

Ele geralmente replica os dados.

Replicação => Dados são copiados.

Partição => Divide os dados na rede. Ex: Dados de nota fiscal do RS fica em um Cluster e a do RJ fica em outro Cluster. Para que querer as notas fiscais do RS, só acesso um Cluster.

Mutation => Suporte ao sistema em alterar dados.

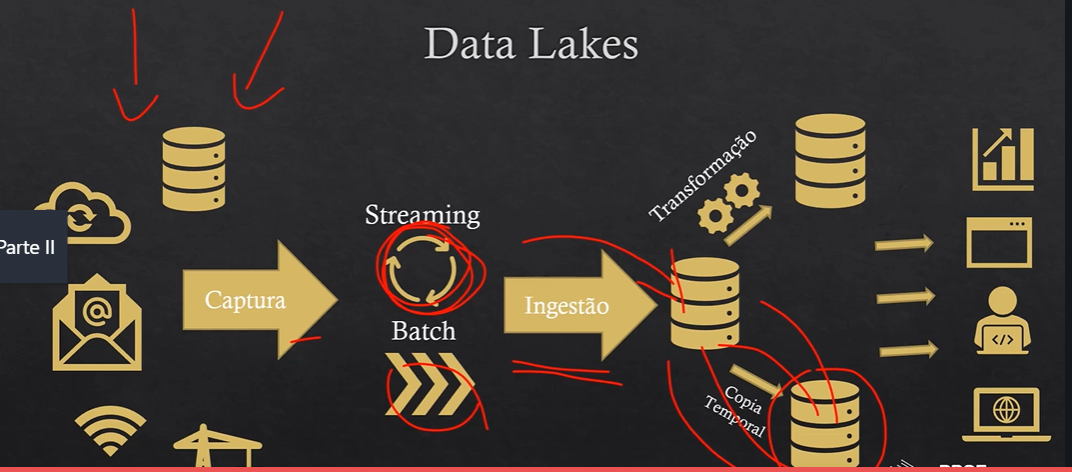
Data WareHouse

Armazém de dados cujo o proposto é armazenar dados dimensional para fins analítico.

Data Lake

Suporta todo tipo dados. É construído em um sistema de arquivos distribuído(não é um banco de dado) , chamado HDFS, que faz parte do Hadoop.

Podem ter uma área de Descarga(dados nativos),Consumo(dados já tratados) e cópias temporais.



Batch vs Streaming vs Interativo

Batch => Processamento dos dados não precisam ser imediatos. Podem levar horas, dia, semanas. Ex: Validar o processamento da Folha de Pagamento.

Streaming => Processamentos dos Dados são em tempo real ou próximo disso. Ex: Sistema de Detecção de fraude ao passar o cartão de crédito.

Event = Dado

Producer = Gerador dos Dados. Passar o cartão de crédito.

Subscriber = Quem Consome. O modelo de machine Learning que vai verificar se é fraude ou não.

Interativo (Ad Hoc) => Processamentos nada casual. Fazer um SQL para extração de dados.

Tipos de Straming

Nativo => Os dados são processados assim que chegam, sem esperados por demais dados.

Micro-Baching => Dados são agrupados e processados em Grupos.

Change Data Capure

Capturar mudanças nos dados.

Tolerância a Folha

Full ->Master e Slace são recuperados

Executer Recover -> Apenas os Slaces.

ETL vs ELT

Etl -> Data WareHouse. Já faz a

Elt -> Data Lake. Apenas , conforme a demanda, ele faz a transformação.

Data at Rest => Dados que estão parados, descansando.

Data at Wire => Dados estão sendo transmitidos.

Teorema de Cap =>

Para Designer de Sistemas Distribuídos.

Segundo o Teorema de Cap, é impossível ter mais de 2 características:

Consistência => Todos os nós da Rede retornam a mesma versão dos dados.

Disponibilidade => Todos os nós sejam disponibilizados para leitura e escrita em tempo razoável.

Partição de Tolerância a Falha => Sistema seja tolerante a falhas .

Em caso de falha de repartição, o que você prefere? Que apareça a informação, mesmo que a informação possa estar errada, ou que não apareça nenhuma informação?

Sistema Financeiro => Não apareça nenhum dados.

Sistema de Busca => Pode aparecer os dados antigos.

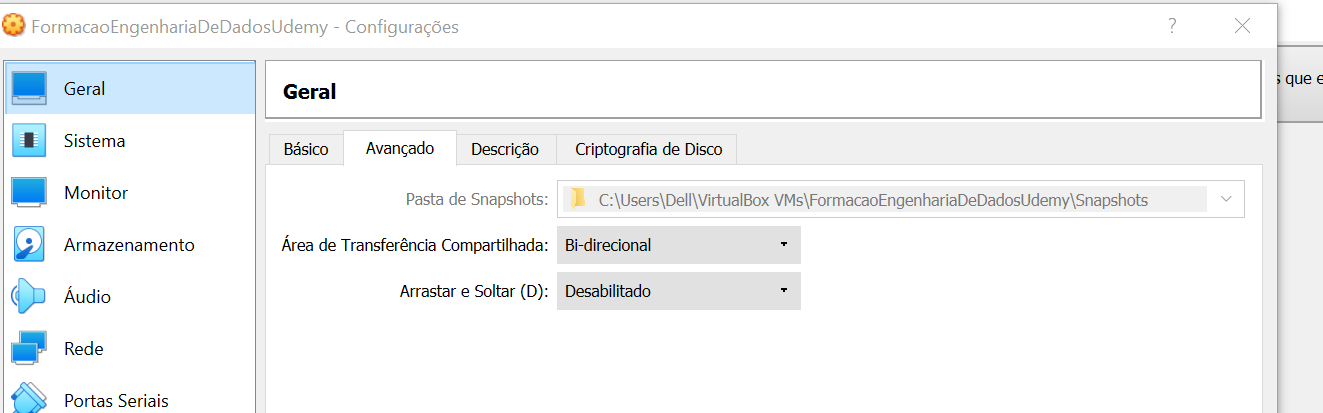
Parte Prática:

Vamos baixar uma máquina virtual:

Oracle Virtual Box => Instalar o Oracle Virtual Box.

cloudera quickstart vm => S.O Linux que já roda uma distribuição do Hadoop instalado.

Máquina => Configurações.



# Módulo 2 – Modelos Relacionais

ACID

Atomicidade => Ou a transação será executada totalmente ou não será executada.

Consistência => A transação cria um novo estado válido dos dados ou em caso de falha retorna todos os dados ao seu estado antes que a transação foi iniciada.

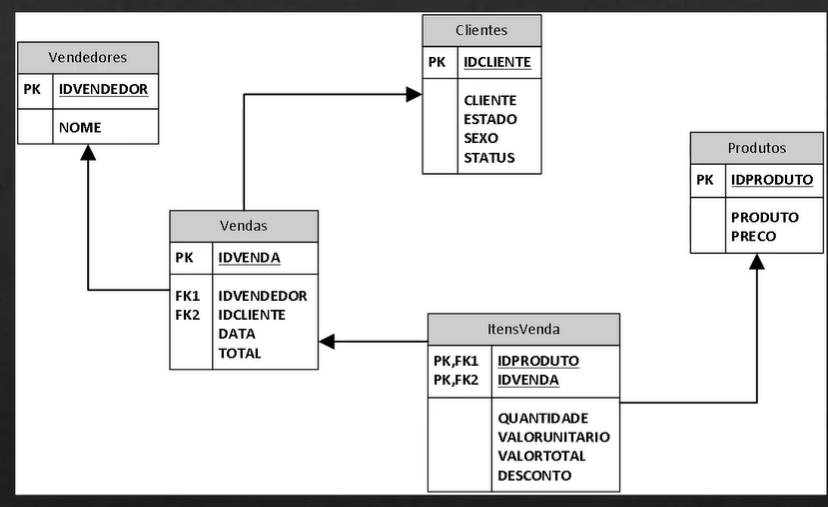
Isolamento => Uma transação em andamento mas ainda não validada deve permanecer isolada de qualquer outra operação, ou seja, garantimos que a transação não será interferida por nenhuma outra transação concorrente.

Durabilidade => Dados validados são registados pelo sistema de tal forma que mesmo no caso de uma falha e/ou reinício do sistema, os dados estão disponíveis em seu estado correto.

Formas Normais

1° => Todos os atributos de uma tabela devem ser atômicos, ou seja, a tabela não deve conter grupos repetidos e nem atributos com mais de um valor.

2° => Todos os atributos não chaves da tabela devem depender unicamente da chave primária.



Após a criação da máquina Virtual, segue os seguintes passos para poder criar um banco PostGree:

#apagamos dados do yum

cd /etc/yum.repos.d/

sudo rm \*.\*

#baixamos o RPM

cd ~

sudo yum install http://download.postgresql.org/pub/repos/yum/reporpms/EL-6-x86\_64/pgdg-redhat-repo-latest.noarch.rpm

#editamos o arquivo de configuração

sudo gedit /etc/yum.repos.d/pgdg-redhat-all.repo

#atualizamos Yum

sudo yum update

#agora é só executar os demais comandos normalmente:

#instalar pacotes cliente

sudo yum install postgresql11

#instalar pacotes servidor opcionais

sudo yum install postgresql11-server

#inicializar e definir inicio automatico

sudo service postgresql-11 initdb

sudo chkconfig postgresql-11 on

sudo service postgresql-11 start

#logar no postresql e criar banco de dados

sudo -u postgres psql

# Criar um banco de Dados no POstgree

Create dataabse ed;

#listar banco de dados

\list;

# Mudar o banco de dados

\c ed;

# Iniciar a execução dos Scripts 1 até 6

\i /home/Cloudera/Downloads/xxxxxx.sql

#Sair

\q

# Módulo 3 – Modelos Dimensionais

DataWareHouse clássico => Serve para consolidar os dados para gerar dados analíticos.

Operações de Inclusões e Consulta.

Conjunto de DataMarts

Data Mart => Banco de Dados analíticos por departamento.

Olap => São sistemas analíticos que serve para manipular e analisar grande volume de dados sob múltiplas perspectivas.

Diferença entre Modelo Relacional vs Dimensional



Fez uma alteração no campo Status, no modelo Dimensional inseriu uma nova linha.

Granulidade => Nível de detalhamento das informações. Ex; Se é por estado ou cidade.

Cubo => Permite aumentar ou reduzir o nível de detalhe da informação, aplicar filtro, agrupar, etc..

Dados Multidimensionais => São dados que se podem ver em várias perspectivas, de forma dinâmica, calculadas de forma automática., conforme perspectiva selecionadas.

Tipos de DRILL

Drill Down => Mais Detalhe

Drill Up => Menos Detalhe

Drill Across => Pula de um nível hierárquico para outro.

Slice => Filtro. Fateamento.

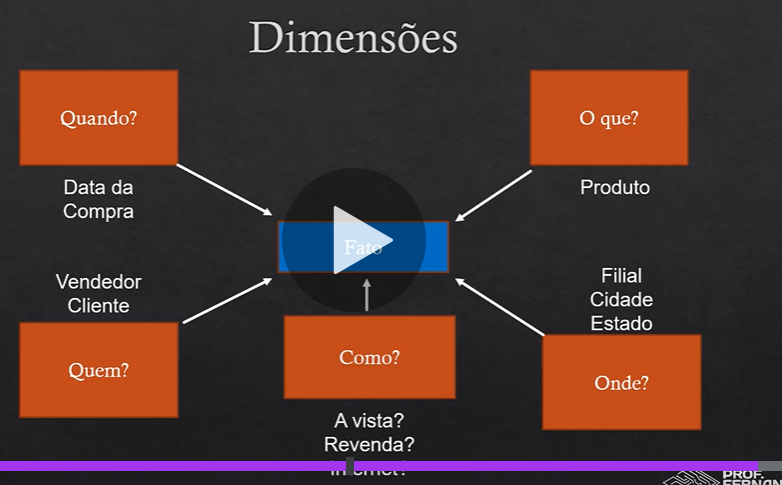
Dice => Mudança de perspectiva.

Fato vs Dimensão vs Medidas

Fato => Informação central. Informação que quer analisar.

Dimensão => As perspectivas que quer analisar as informações centrais.

Medidas => Relacionados ao Fato. São os valores, métricas.



Dimensão Tempo é obrigatória.

# Módulo 4 – NoSQL

Resolver problemas mais específicos.

Tipos Principais:

Key-Value DataBase: Redis, Amazon DynamoDB.

Colunas Ordenadas: BigTable(google). Apache HBASE. Hypertable. Cassandra.

Banco de Dados de Documentos: Suporte JSON. MongoDB.

Grafos: Redes Sociais. Neo4j.

# MongoDB.

|  |  |
| --- | --- |
| Relacional | MongoDb |
| Tabela | Coleção |
| Linha | Documento |
| Coluna | Campo |
|  |  |
|  |  |

Comando:

#Criar um banco

use dbmidias

#Criar uma coleção

Db.CreateCollection(“clientes”);

#Criar uma Linha

Db.posts.insert({nome:”José”,postagens:”Bons Produtos”})

#Inserir mais de uma linha

Db.posts.insert({nome:”José”,postagens:”Bons Produtos”}, {nome:”José”,postagens:”Bons Produtos”}, {nome:”José”,postagens:”Bons Produtos”})

#Recuperar Documentos

Db.clientes.find() //Traz todas as linhas da collection Clientes

Db.clientes.findOne() //Traz a primeira linha da collection

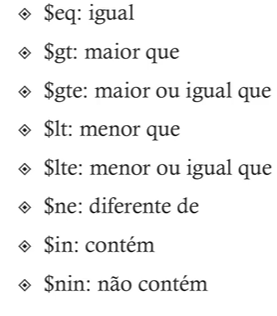
Db.clientes.find().pretty() // Formata o documento para visuzalição.

#Documento Específico

Db.posts.find({nome:”José”})

Db.posts.find({nome:”José”, postagem:”Bons Produtos”})

#Operadores



#idade menor ou igual a 12

Db.posts.find({postagens:”Produtos caros”,idade:{$lte:12}})

#Condição OR

Db.posts.find($or:[{nome:”José”},{nome:”Antonio”}])

# IN

Db.posts.find({postagens:{$in:[“Produtos caros”,”Loja Suja”]}})

#Projeções => São equivalentes a quais campos do SELECT que a gente quer trazer. Quando colocamos 0, não é para trazer. Quando colocamos 1, é para trazer.

Db.posts.find({nome:”José”},{\_id:0,nome:0})

#Limite

Db.posts.find().limit(2)

#Ordenar => 1 - Crescente -1 - Decrescente

Db.posts.find().sort({nome:1})

#Concatenar Funções

Db.posts.find().sort({nome:-1}).limit(2)

#Atualizar

Update()

Db.posts.update({nome:”André”},{$set:{idade:29}})

Save() => Ou atualiza ou Insere.

#Delete o Documento

deleteOne()

db.posts.deleteOne({nome:”André”})

deleteMany() => Exclui todos os documentos conforme critério

remove() => Exluir todos os documentos da coleção.

#Exlcuir a Coleção

Db.posts.drop()

#Outros Comandos

Db => Verifica o banco que está sendo usado.

Show Dbs

Db.dropDataBase()

Show collections

#Backup => mongodump (shell do linux)

Sudo mkdir /home/Cloudera/bkpmongo

Sudo mongodump –out /home/Cloudera/bkmongo –db dbmidias

–db dbmidias (Qual o banco do BackUp)

#Restaurar => mongorestore

Mongo Shell

Use dbmidias

Db.dropDatabase()

Db.posts.find() //Não vai achar

Shell Linux

Sudo mongorestore /home/Cloudera/bkpmongo

Mongo Shell

Use dbMidias

#Exportar os Dados

Sudo mkdir /home/cloudera/exportmongo

Sudo mongoExport –db dbmidias –collection posts –out /home/cloudera/expostmongo/posts.json

#Importar Dados

Mongoimport –db dbmidias –collection posts –file /home/Cloudera/downloasds/post.json.

Pode ser que aqui tenha que acrescentar o –legacy no final

#Instalação do Mongo

cd /etc/yum.repos.d

Insirir o arquivo xxx nessa página que é referência do yum para instalar o mongo.

Instale

sudo yum install -y mongodb-org

Inicie o serviço

sudo service mongod start

Configure reinicio automático

sudo chkconfig mongod on

Acesse o Shell

mongo

# REDIS

Banco de Dados Chave-valor em memória.

Banco de Dados em Memória

Pode usar disco para persistência.

Aplicações:

Cache de Sessões : E-commerce contendo seção que armazena o carrinho de compras.

Cache de Páginas : Páginas mais acessadas podem ser armazenadas em memória, não sendo preciso renderizar a página para a cada nova requisição.

Suporte a Cluster e Streaming.

Suporte de Dados:

* String => SET chave valor. GET chave.

Ex: SET 1 “Engenharia de Dados”.

GET 1

#Incluir vários conjuntos de chave/valor

MSET 1040 ANALISTA 1050 GERENTE 1060 TESTADOR

#Verifica se uma chave existe. Retorna 1 ou 0

EXISTS 1

# Excluir um registro. Retorna 1 ou 0

DEL 1

#Verificar o tipo da Chave

TYPE 1

#Expiração

SET 1 “Engenharia de Dados” EX 60 NX

EX: Segundos

PX: Milesegundos

NX: Só funciona se a chave não existe

XX: Só funciona se a chave já existe.

#Definição Posterior da Expiração.

(EXPIRE or PEXPIRE) 1 60

#Verificando o Tempo para Expirar

PTTL 10 -> Verificar o tempo em Segundos

TTL 10 -> Verificar o tempo em Milesegundos.

#Remove o tempo de Expiração => Em memória definitivamente.

Persist 10

# GETRANGE

Set 1020 “Engenheiro de DAdos”

GETRANGE 1020 0 9

’Engenheiro’

# Atualizar Dados

GETSET 1030 “Cientista de Dados”

# Retorna o valor da variável

MGET 1020 1030

1. “Engenheiro de Dados”
2. “Cientista de Dados”

#Tamanho do valor

STRLEN 1020

* Hashes=> Conjuntos de Campos/Valores, associados a uma chave. Sintaxe: HMSET chave campo, valor, campo, valor ....

Ex: HMSET CADASTRO NOME JOSE PROFISSAO ENGENHEIRO CIDADE “SANTA MARIA”

#Excluir Chave/Valor

HDEL CADASTRO CICADE

#Retornar valor de um Registro

HGETALL CADASTRO

#Recupepar Valores de Campos

HMGET CADASTRO NOME PROFISSAO => Vai trazer o valor dos campos Nome e Profissão.

# Recuperar todos os Campos

HVALS CADASTRO

# Verifica se Existe Chave => Retorna 1 ou 0

HEXISTS CADASTRO NOME

# Retorna Número de Campos

HLEN CADASTRO

#Retorna Lista de Campos

HKEYS CADASTRO

* Lists => Uma lista de textos associados a uma chave.

#Inclusão no início da Lista

LPUSH 4545 SQLSERVER ORACLE POSTGRES

#inclusção no final da lista

RPUSH 3535 DB2

#Retornar Intervalo

LRANGE 4554 0 3

# Insere antes ou depois de Valor

LINSERT 4545 AFTER(BEFORE) ORACLE FIREBASE => Vai inserir Firebase

#Atualizar

LSET 4545 1 POSTGRESQL => Comando chave posição Novo\_Valor

#Retorna valor pelo Índice

LINDEX 4545 1 => Comando Chave Posição

# Comprimento da Lista

LLEN 4545

#Removendo

LPOP (RPOP) 4545 => Remove do inicio e do fim da lista.

* Sets => Coleção não ordenada, não repetida.

#Adicionar

SADD 12 HADOOP SPARK HIVE PING

SADD 12 FLUME OOZIE SOLR

SADD 12 FLUME => Vai dar erro porque já existe um Flume no Sets

#Recuperando todos os Membros

SMEMBERS 13

#Recuperar o número de Membros

SCARD 13

#Verifico se um valor é membro

SISMEMBER 13 HADOOP

#Remove um ou mais membros

SREM 13 OOZIE SPARK

#Diferença entre dois Conjuntos

SDIFF 13 14

#Igualdade entre dois Conjuntos

SINTER 13 14

#----Ordenados por meio do Score

# Adicionar

ZADD 35 0 NOSQL -> Comando chave score valor

ZADD 35 4 ORACLE

#Índice de um membro

ZRANK 35 DIMENSIONAL

#Conta membros com Scores entre...

ZCOUNT 35 0 5

#Retorna Membros

ZRANGE 35 0 5

ZRANGE 35 0 5 WITHSCORES -> Traz os scores vazio.

#remove

ZREM 35 DIMENSIONAL

Transações

Multi -> É o Begin

Exec -> É o commit.

Discard -> É o RollBack.

Instalação

wget http://download.redis.io/releases/redis-5.0.5.tar.gz

tar xzf redis-5.0.5.tar.gz

cd redis-5.0.5

make -> Comando para compilar o programa

#inicia serviço

src/redis-server

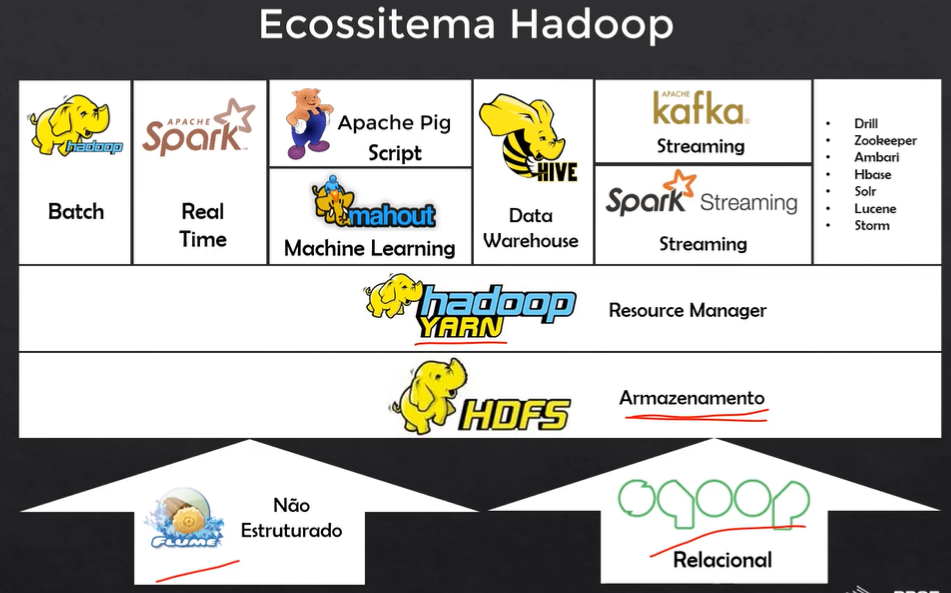
#cliente

src/redis-cli

#ping

Retorna pong

# MÓDULO 7 – Processamento em Batch com Hadoop



Ingestão de Dados (Importar os Dados)

Flame -> Dados não estruturados

Sqoop -> Dados Relacionais. Banco de Dados Relacionais

HDFS -> sistema de Arquivos distribuídos

YARN -> Gerenciamento dos Recursos. Gestão dos Recursos

HADOOP -> Aplicação para processar grande quantidade de Dados em lote(batch)

Apache Spark -> Aplicação em processamento de dados em tempo real em memória

Apache Ping ->Aplicação de Script. Desenvolver Script para realizar aplicação MAP/REDUCE.

Map/Reduce

Dividir Tarefas de processamento de dados em vários nós(servidores).

Dados são dividios em blocos.

Divisão de grandes problemas em pequenas tarefas.

Mapeamento é executado em paralelo em nós.

Após o mapemaneto, inicia-se o a Redução.

Suffle -> Fase intermediária.

Mahout -> Biblioteca de Machine Learn

Hive -> Solução de DW para Big Data.

Kafka Straming, Spart Straming -> Para Straming.

HADOOP

Master vs Slave

Master -> Faz a Gestão completa. Contém Logs.

NameNode -> Gestão do HDFS em um nó. Log, metadados, copia, encontra , exclui arquivos.

JobTracker -> Distribui as tarefas entre nós.

TaskTracker -> Recebe as informações do JobTracker. Agendamento, Balanceamento de Carga, gestão de falhas.

Slaves (DataNodes) -> Mantém os dados, replica os dados para caso um nó caia.

DataNodes são configurados como Ativos e StandBy.

YARN

Alocação de Recursos de forma global.

Priorização das tarefas

Tolerância a falhas

Componentes:

Container -> Mantém as tarefas

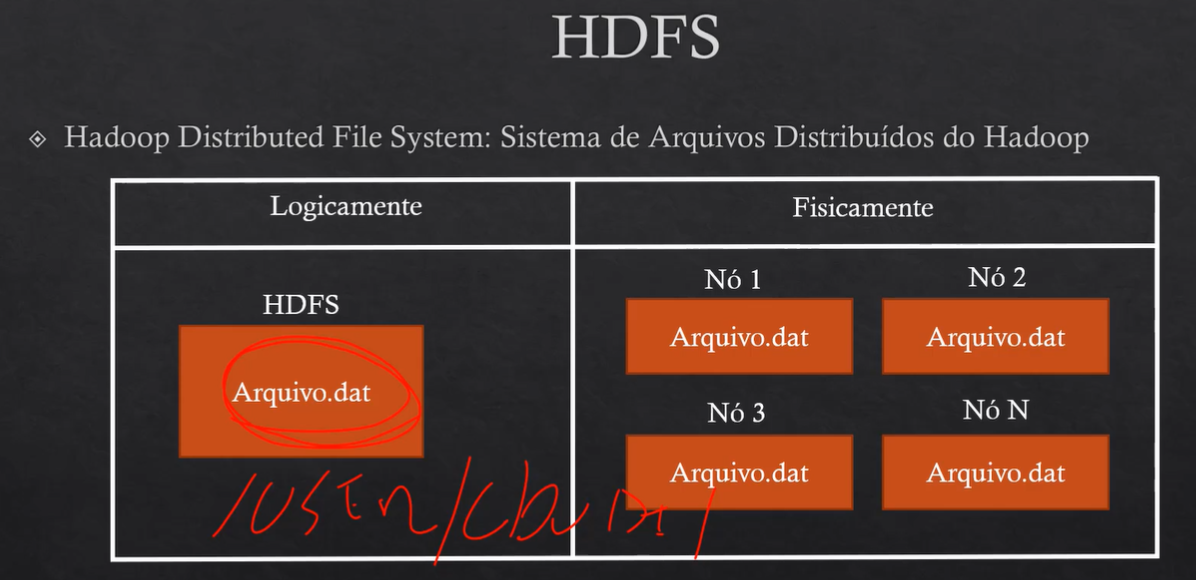
ResourceManager -> um por cluster

AplicationManager -> Gerencia Atividades, otimização, distribuição de recursos.

NodeManager -> um por nó. Execução dos nós

Aplication Master -> Distribuir tarefas no Container.

HDFS



Sistema de Arquivos Distribuídos do Hadoop.

Arquivo Separado em Blocos que armazenam dados.

Distribuídos em nós de redes.

Cópias replicadas. Replicação dos dados. Default 3 nós.

Principais Comandos do HDFS

#Listar Conteúdo de diretório

Hdfs dfs -ls -R / -> Listagem Recursiva. O “/” é o diretório. É o root do HDFS.

Hdfs dfs -ls -d / -> listagem simples

#Copiar Arquivos do Sistema Local para o HDFS

Hdfs dfs -put /user/arqu.txt /delta/arq.txt -> Primeiro é o arquivo do Linux e o segundo é o do HDFS

#Mover Arquivos do Sistema Local para o HDFS

Hdfs dfs -mv input.txt /user -> copiando para dentro da pasta user do HDFS

# Remove Arquivo ou Pasta

Hdfs dfs -em /user/input.txt -> usa -r para forma recursiva.

#verifica o tamanho do arquivo

Hdfs dfs -du input.txt /user

#Exibe o conteúdo do Arquivo

Hdfs dfs -cat /user/input.txt

#Cria/Remove uma pasta

Hdfs dfs -mkdir(rmdir) /user/diretório

#mostra a parte final de Arquivo

Hdfs dfs -tail /user/input.txt

#conta o número de diretório do Arquivo

Hdfs dfs -count /user

# Altera o fator de Replicação de um Arquivo. Que no padrão é 3.

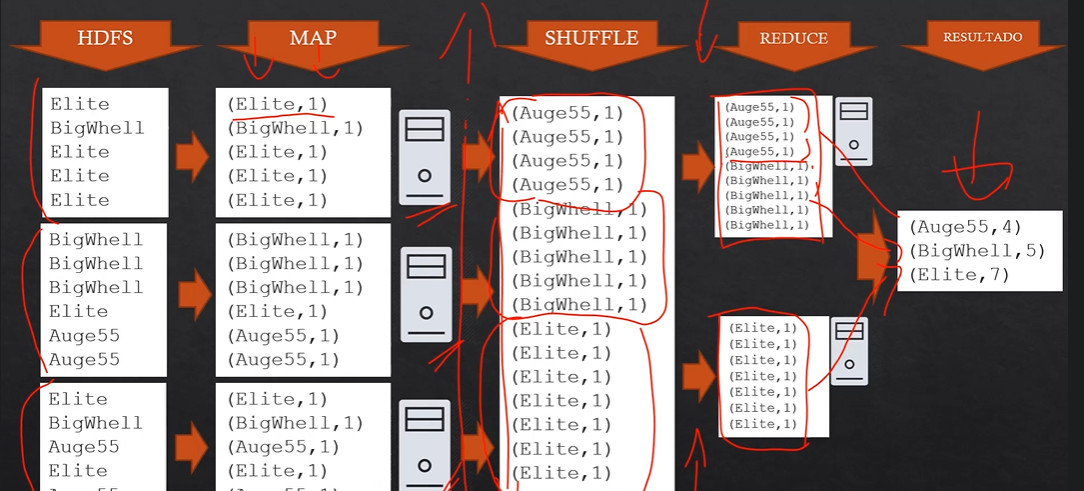
Hdfs dfs -setrep 3 /user/input.txt

# Verifica o Estado do Sistema de Arquivos

Hdfs -dfs -fsck /

#Informa estatística do arquivo ou diretório

Hdfs dfs -stat %f /user/input.txt -> %F -> Tipo, %n nome %r fator de replicação. %y data da modificação.



HADOOP não é bom para:

Processamento Real-Time

Pequenos Volumes de Dados

Problemas complexos(grafos)

Operações que precisam grande tráfego de dados na rede

HADOOP é insuperável para:

Garande volumes de dados em processamento de batch.

**“Hello Word” do Hadoop**

cd /home/cloudera/Downloads

#descompatamos o arquivo

unzip hadoop.zip

#vizualizar

cat pesquiza.txt

cat WordCount.java

#criar pasta no hdfs

hdfs dfs -mkdir /contar/

#copiar o arquivo para a pasta criada

hdfs dfs -put pesquisa.txt /contar/pesquisa.txt

#Criar o .JAR da classe do arquivo Java.

#rodar map reduce

hadoop jar /home/cloudera/MRProgramDemo.jar PackageDemo.WordCount /contar/pesquisa.txt /contar2

hdfs dfs -ls /contar2

hdfs dfs -cat /contar2/part-r-00000

# Módulo 8 – SQOOP, HIVE e PING

SQOOP -> Ingestão de Dados Relacionais.

Importa os dados para:

HIVE, HBASE, ACCUMULO, HDFS.

HIVE (Coméia)-> DW. Armazem de Dados. DataWareHouse.

PING -> Script de Programação.

SQOOP ->

Importa 4 tarefas em paralelo, mas precisa de uma chave primária para dividir as tarefas.

A tabela é importada para uma pasta de mesmo nome dentro do HDFS.

Modo Incremental:

Append => Inserir novos registros

LastModified => Alterar os Registro existentes.

Preparar Ambiente:

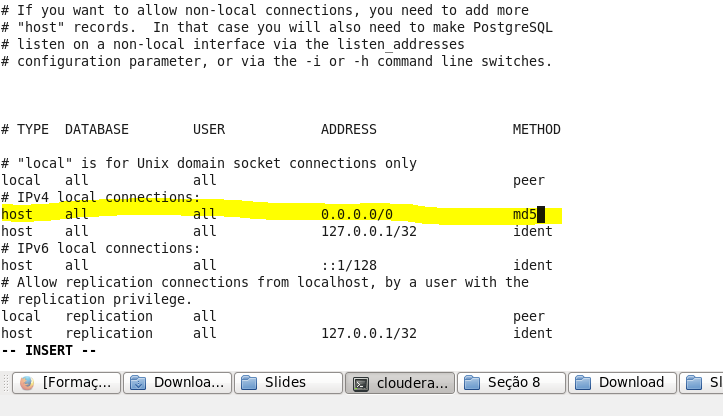
#PostGree

#Permitir acesso externo

Sudo vi /var/lib/pgsql/11/data/pg\_hba.conf

#Incluir a Linha, antes do Ident

Host all all 0.0.0.0/0 md5



#Reinicar o postGree

sudo service postgresql-11 stop

sudo service postgresql-11 start

#Baixar o drive JDBC para o SQOOP

cd /home/cloudera/Downloads

wget http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar

sudo cp postgresql-9.2-1002.jdbc4.jar /var/lib/sqoop/postgresql-9.2-1002.jdbc4.jar

#Rodando o SQOOP

#Listar o banco de dados

sqoop list-databases --connect jdbc:postgresql://127.0.0.1/ --username postgres --password 123456

#Listar as Tabelas

sqoop list-tables --connect jdbc:postgresql://127.0.0.1/ed --username postgres --password 123456 -- --schema dimensional

#Importar as tabelas

sqoop import-all-tables --connect jdbc:postgresql://127.0.0.1/ed --username postgres --password 123456 -- --schema dimensional

#verifica a importação

sudo hdfs dfs -ls /user/cloudera

#abre um arquivo

sudo hdfs dfs -cat /user/cloudera/fatovendas/part-m-00000

HIVE -> DataWareHouse com Hadoop

Fisicamente, o Hive fica em /user/hive/warehouse.

Preparando o ambiente:

Beeline -> Conectar ao Hive

!connect jdbc:hive2://

Show databases;

Create table ed;

Use ed;

Criar as tabelas .....

show tables;

# MÓDULO 9 – PING e FLUME

Ping -> Abstração do MapReduce. Linguagem de Alto nível. Ping Latin. Parecido com Hive.

Não é um DataWareHouse, igual ao Hive.

Tuple -> Registro

Bag -> Conjunto de Registros.

O Hive coloca os dados em disco. O Ping é em memória.

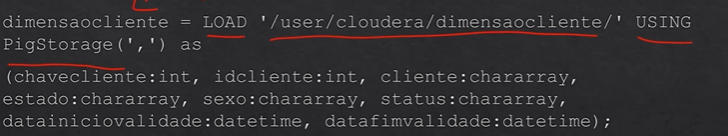
Pig -x local -> Abre o CMD do ping em Local

PIG -x mapreduce -> Abre o CMD do pig em MapReduce

Pig -f local ‘/user/scrippt.ping’ -> executar o Script.

Comandos:

#Carregar dados do HDFS



#Visulizar os Dados

Dump dimensaocliente

#Limitar os dados

Dim10 = limit dimensaocliente 10;

#Agrupar

Clienteagrupado = GROUP dimensaocliente by status;

#Junção

Juncaocliente = JOIN fatovenda by chavecliente, dimensaocliente by chavecliente;

#Filter

dimensaoclienteFilter = FILTER dimensaocliente by status==’platium’;

#Dividir

SPLIT dimensaocliente into dimensaoGold if status==’Silver’, dimensaoPlatium if status==’Platium’;

#Ordenar

dimensaoCliente = ORDER dimensaocliente by cliente ASC;

#Persistir os dados

STORE dimensaoclientePlatium into ‘/user/Cloudera/ping/’ USING (‘,’);

FLUME

Ingestão de Dados não estruturados.

Suporte de múltiplos tipos de origens e destinos.

Tolerante a falhas.

Permite a construção de fluxos de dados.

Caracteristicas:

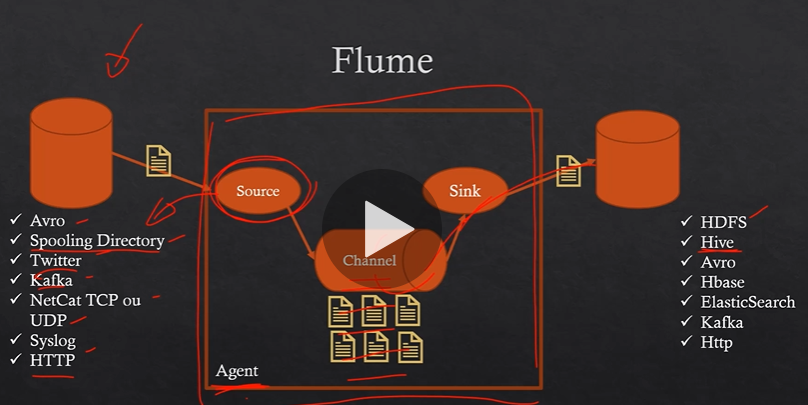
Event -> Unidade de dados.

Agent -> Processo do consumo dos dados.

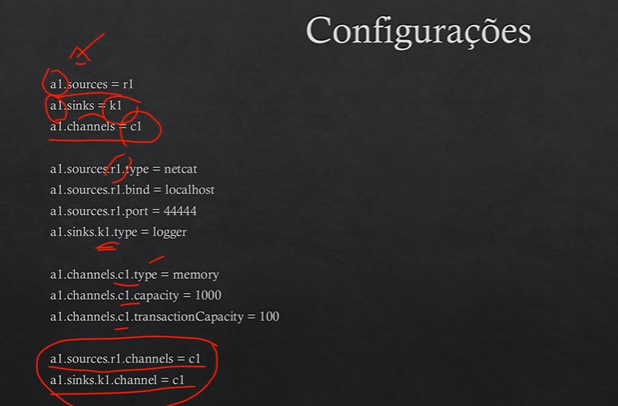
Source -> É origem dos dados (Twitter, Kafka, Http)

Channel -> Fila onde os dados aguarda.

Sink -> Coloca no destino final. (HDFS,Hive,HBase, ElasticSearch,...)



#Configurações:



#Iniciando um agente

Flume-ng agent

Flume-ng agent –conf conf –-conffile conf.conf (Arquivo de Configurações) –name al – Dflume.root.logger=INFO,console

IMPLEMENTAÇÃO.

Monitorar uma pasta e jogar os dados no HDFS para cada alteração nessa pasta.

#1 – Criar uma pasta no Linux

Mkdir /home/cloudera/flume

#2 Criar um Diretório no HDFS

Hdfs dfs -ls /user/flume

#3 Criar um arquivo .config do Flume

#4 Executar o agent

sudo flume-ng agent –conf conf –conf-file /home/cloudera/example.conf -–name a1 (Nome do Agente) – Dflume.root.logger=INFO, console;

# Módulo 10 – Processamento Real Time e Streaming com Spark.

Spark -> Ferramenta de processamento de dados.

Compatível com o Hadoop.

Persiste os dados em memórias, mas pode ser em disco.

O Shell é em Scala ou Python

Componenetes:

Core -> Gestão de Recursos, tolerância a falhas, monitoriamento e agendamento.

SQL -> consulta SQL. Suporte ao HIVE.

Streaming -> Processamento de Dados em Tempo Real

Mllib -> Biblioteca de Machine Learn

GraphX -> Processamento de grafos.

Modos

Standalone e Cluster

Spark vs Hadoop

Hadoop -> Processamento em batch

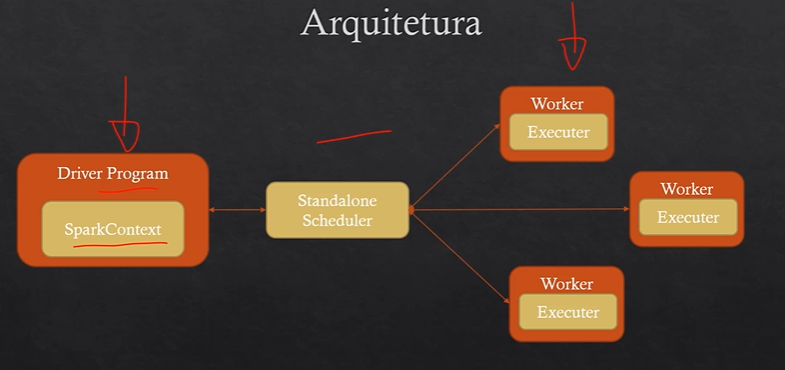
Spark -> Tempo Real.

Memórias física é mais barata que a memória Hom.

RDD -> Dados imutáveis(dados que não mudam), distribuídos pelo cluster

Operações sobre um RDD cria um novo RDD, por isso que não muda os dados.

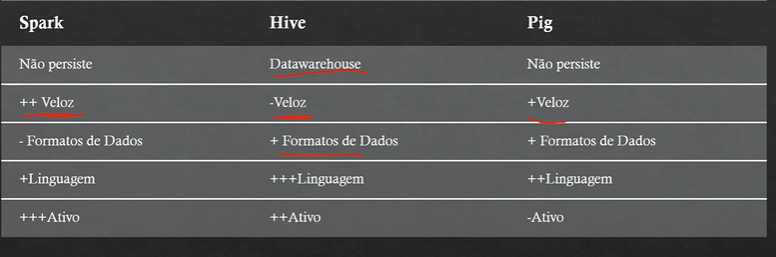
SparkContext -> Conexão transparente com o Cluster.



Shell:

Scala: spark -shell

Python:pyspark



Investir mais em Spark, Hive e Hadoop. Pig nem tanto.

RDD

Podemos persistir o RDD em memória:

RDD.Persist

Memory\_only

Memory\_only\_ser (o ser é em binário, mais compacto)

Memory\_and\_disk

Memory\_and\_disk\_ser

Disk\_only

Operações:

Transformações => Geram um novo RDD. Ex: Filter, map, sample, distinct, subtract, cartesian, keys, values, e etc....

Ações => Cálculo exibido no console. Não gera um novo RDD.

Ex: Collect() => Retorna todo o RDD



#prática:

Pyspark

Sc => variáel de contexto.

numero = sc.parallelize([1,2,3,4,5,6,7,8,9,10])

numero.first()

numero.top(5)

numero.collect() => Meu RDD

numero.persist(StorageLevel.MEMORY\_ONLY)

numero.unpersist()

fitro = numero.filter(lambda filtro: filtro >2)

filtro.collect()

amosta = numeros.sample(True,0.5,1)

amostra.collect()

mapa = numeros.map(lambda mapa:mapa\*2) => multiplicar cada número por dois.

Mapa.collect()

Compras = sc.parallelize([(1,200),(2,300),(3,120),(4,250),(5,78)])

Chaves = Compras.keys()

Chaves.collect()

Compras.countByKey()

Soma = compras.mapValues(lambda soma:soma+1)

Soma.collect()

#pratico 2

#Copiar o arquivo de configuração de conexão do HIVe para o Spark

Sudo cp /usr/lib/hive/conf/hive-site.xml /usr/lib/spark/conf/

#Conexão Spark e Hive

Pyspark

From pyspark.sql import HiveContext

Contexto = HiveContext(sc)

Banco = contexto.table(“ed.des\_vendas”)

Banco.show()

Banco.registerTempTable(“des\_vendas”)

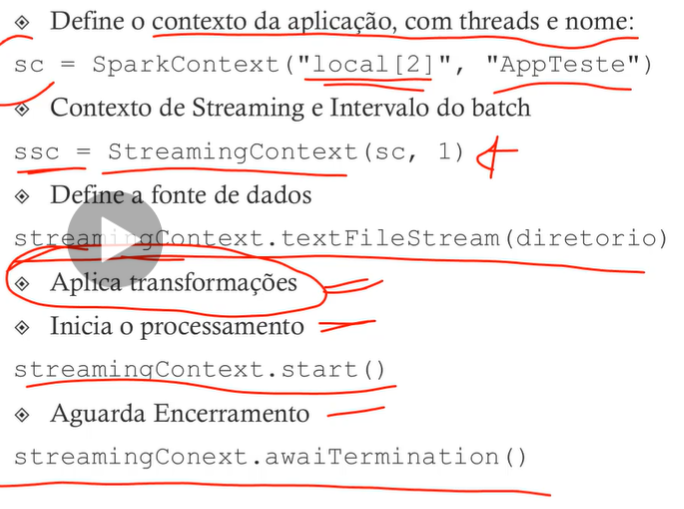
Context.sql(“select \* from des\_vendas”).show()

Spark Streaming

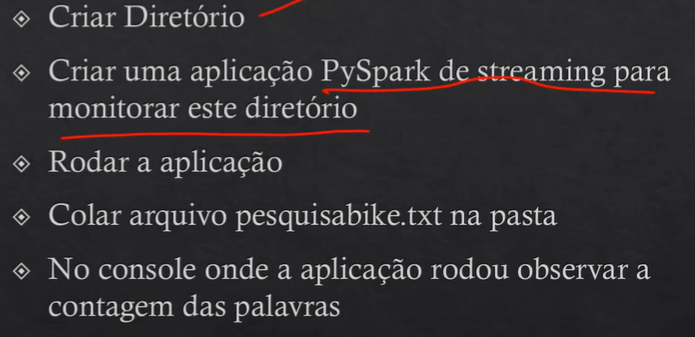
Em tempo real , ou próximo ao tempo real.



Dstream => Representação interna dos dados.



#Prática -> Ler aquivo no diretório. Um mesmo arquivo não é lido novamente, mesmo que modificado. Deve ser lido um mesmo arquivo ou renomear.



from pyspark import SparkContext

from pyspark.streaming import StreamingContext

#threads e nome no contexto

sc = SparkContext("local[2]", "Contagem")

#contexto e intervalo do batch

ssc = StreamingContext(sc, 10)

#pasta a monitorar

pesquisa = ssc.textFileStream("file:///home/cloudera/spark/")

#contagem

contagem = pesquisa.flatMap(lambda palavra: palavra.split(" "))

contagem = contagem.map(lambda pal: (pal, 1))

contagem = contagem.reduceByKey(lambda a, b: a + b)

#imprime

contagem.pprint()

ssc.start()

ssc.awaitTermination()

#Executar o monitoriamento

spark-submit /home/cloudera/exemplo.py

#inserir um arquivo de texto na pasta spark, que está sendo monitorada.

#Ao inserir, vai executar.

# Módulo 11 – Aspectos Gerais sobre Engenharia de Dados

Data Lake

Data Lake = Armazém corporativo de dados

Versão “Big Data” do DW

Normalmente em HDFS

Dados estruturados/não estruturados/semi-estruturados.

Estrutura :

Landing Layer = Dados brutos

Analytic Layer = Dados Tratados para Consumo

Arquivamento = Dados Históricos.

# Módulo 13 – Linux

