Big Data & Hadoop

Sadi Júnior, Parasita

Universidade Federal de Santa Catarina

Cloudera + Hadoop

O ambiente de desenvolvimento utilizado no projeto foi um contêiner do Docker contendo as VMs Cloudera QuickStart, Apache Hadoop e Cloudera Manager, entre outros.
Para maiores informaões, acesse:
`https://hub.docker.com/r/cloudera/quickstart/`

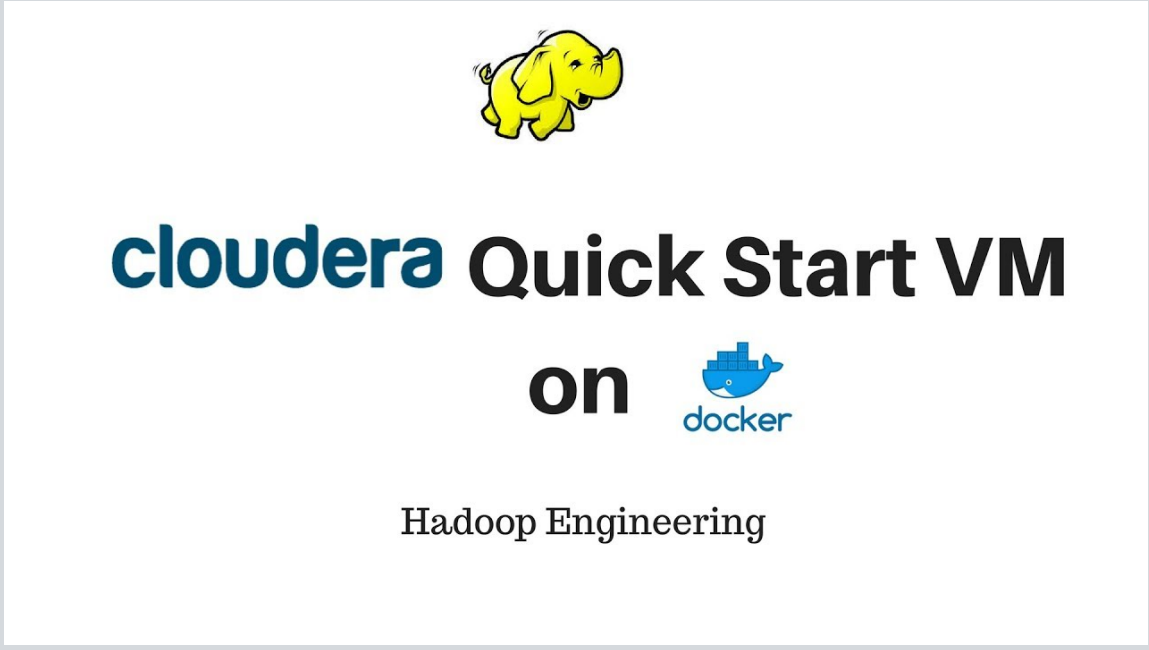

Figura 1: *Cloudera + Hadoop*

Dataset

O *dataset* utilizido pode ser acessado em `https://brasil.io/dataset/gastos-deputados/cota_parlamentar`.
Trata-se de um *dataset* que contem os gastos parlamentares registrados na Câmara dos Deputados.
O mesmo consiste em um arquivo csv, cujo separador é o carácter “,”.

Dataset

Tal *dataset* possuí os campos:

- codlegislatura: inteiro, identifica a legislatura do parlamentar.
- datemissao: data de emissão do gasto.
- idedocumento: inteiro.
- idecadastro: inteiro.
- indtipodocumento: inteiro.
- nucarteiraparlamentar: inteiro.
- nudeputadoid:inteiro.
- nulegislatura: inteiro.
- numano: texto, indica o ano do gasto.

Dataset

- numespecificacaosubcota: inteiro.
- numlote: inteiro.
- nummes: texto, identifica o mês do gasto.
- numparcela: inteiro, identifica qual a parcela sendo paga.
- numressarcimento: inteiro.
- numsubcota: inteiro.
- sgpartido: texto, identifica o partido do parlamentar.
- sguf: texto, identifica a UF do parlamentar.
- txnomeparlamentar: texto, o nome do parlamentar.
- txtcnpjcpf: texto, o CPF ou CNPJ usado para compra do produto ou serviço.

Dataset

- txtdescricao: texto, descreve o produto ou serviço comprado.
- txtdescricaoespecificacao: texto, mais especificações sobre o produto ou serviço.
- txtfornecedor: texto, identifica o fornecedor do produto ou serviço.
- txtnumero: inteiro.
- txtpassageiro: texto.
- txttrecho: texto.
- vlrdocumento: inteiro.
- vlrglosa: inteiro.
- vlrliquido: real, o valor líquido do produto ou serviço.
- vlrrestituicao: real, o valor restituído.

Jobs e Índices

Foram implementados, ao todo, 5 *jobs* de mapeamento e redução. Cada um com uma funcionalidade:

1. Gastos dos parlamentares:
   Mapeiamento de todos os parlamentares e seus respectivos gastos e posterior cálculo do total gasto pelo mesmo ao longo de todo o período, além do cálculo dos valores máximo e mínimo gastos pelo parlamentar e a média de quanto o mesmo gasta. Utiliza o nome do parlamentar como chave[1] e o custo líquido como valor.

[1] na versão inicial utilizava nome e CPF/CNPJ porém, dada a quantidade absurda de entradas diferentes de CNPJs para o mesmo parlamentar, afora a quantidade absurda de entradas com esse campo vazio, se optou por utilizar apenas o nome do parlamentar como índice

Jobs e Índices

2. Gastos por produto/serviço:
   Cálculo do total e da média dos gastos dos parlamentares por produto/serviço durante todo o período. Sendo oposto ao primeiro *job*, utiliza uma chave composta pelo identificador do parlamentar (CPF/CNPJ), nome do mesmo e nome do produto/serviço comprado, e tem o custo líquido como valor.
3. Gastos do partido por ano:
   Gasto total de cada partido por ano. Também utiliza chave composto pelo nome do partido e o ano, e tem como valor o custo líquido.

Jobs e Índices

4. Gastos do governo em produtos/serviços:
   Total do custo de cada produto/serviço durante todo o período. Tem por chave o nome do produto/serviço e por valor o custo líquido do mesmo.
5. Gastos dos parlamentares por período:
   Gastos de cada parlamentar por mês e ano, podendo ser usado para realizar predições de gastos futuros. Tem chave composta (CPF/CNPJ, nome do parlamentar, ano e mês) e o custo líquido como valor.

Implementação dos *Jobs*

- *Mappers*:
  Todos os *mappers* possuem uma implementação similar, uma vez que a entrada de cada um deles é a mesma. E, se tratando de um arquivo CSV, basta fazer o *split* pelo carácter "," e, tendo prévio conhecimento dos índices necessários e suas posições relativas no CSV, basta acessá-los e, em certos casos, concatenar índices para gerar uma chave composto.
- *Reducers*:
  Já os *reducers* possuem um pouco mais de trabalho, uma vez que alguns precisam calcular média, máximo e mínimo. Mas, dados os pares de chave-valor já organizadas pelos *mappers*, os *reducers* ficam apenas com a parte de cálculo dos dados.

Implementação dos *Jobs*

O código abaixo apresenta a implementação de um dos *jobs* criados:

```
1  SalesMapper extends Mapper<LongWritable, Text, Text,
2                      DoubleWritable> {
3
4      @Override
5      public void map(LongWritable key, Text value,
6                      Context context) throws IOException,
7                      InterruptedException {
8
9          String[] lines = value.toString().split(",");
10         String product = lines[19];
11         Double cost = Double.valueOf(lines[27]);
12         context.write(new Text(product),
13                      new DoubleWritable(cost));
14     }
15 }
```

Resultados

Alguns dos resultados obtidos, aqui apresentados de forma parcial, foram:

- Os parlamentares gasta mais na divulgação da atividade parlamentar, seguidos de viagens áereas e telefonia. Os menores gastos estão em serviços de pacotes de locomoção, alimentação e hospedagem.
- Os partidos que mais gastaram em 2018 foram o PT, seguido do MDB e do PP. Já o partido que menos gastou nesse ano foi o SDD.
- Os parlamentares que mais gastaram em todo o período foram Edia Lopes, seguido de Wellington Roberto e Silas Câmara. Já os que menos gastaram foram Athos Avelino, João Fontes e Ayrton Xerez.
- Durante todo o período, os maiores gastos em um produto/serviço foram dos parlamentares Beto Mansur (CONSULTORIAS PESQUISAS E TRABALHOS TÉCNICOS), Weliton Prado (DIVULGAÇÃO DA ATIVIDADE PARLAMENTAR) e Wladimir Costa (Emissão Bilhete Aéreo).

Resultados

Demais resultados, assim como os valores exatos dos gastos apresentados nos resultados acima detalhados podem ser acessados através do *link* `https://drive.google.com/drive/folders/1Cx_Xve2rzw3DfSYMjsZoIILXN9EZ3Z1b?usp=sharing`

Faça você mesmo

Para rodar essa aplicação, primeiramente é necessário possuir um ambiente de desenvolvimento com o *Hadoop* instalada e rodando, além de também possuir o *Maven* instalado. Nossa recomendação é a utilização do contêiner apresentado nos *slides* inicias.

Com o ambiente pronto, e tendo acesso ao código fonte da aplicação, basta descompactar a base de dados, que se encontra nos *resources* da aplicação, e importá-la para o sistema de arquivos do *Hadoop*.

Feito isso, basta executar, na raiz do projeto, o comando **mvn clean install**. Esse comando criará um *jar* executável da aplicação. Nesse ponto, basta apenas executar tal *jar* no *Hadoop*, passando como argumentos o diretório no qual se encontra a base de dados que será processada e o diretório no qual os resultados serão escritos.

Obrigado!

UNIVERSIDADE FEDERAL
DE SANTA CATARINA