Projeto Final - Gerência de Grandes Volumes de Dados

Gabriel Sergio Ferreira[1], **Mariana Suarez de Oliveira**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Av. Gal. Milton Tavares de Souza, s/nº - São Domingos – Niterói – RJ - CEP: 24210-346

gs_ferreira@id.uff.br, msuarez@id.uff.br

Abstract. *This study analyzes a dataset of accommodation on the Airbnb platform in London, where various information can be extracted. To make this analysis, an architecture with management tools for large volumes of data was applied, analyzing from n-grams in accomodation names to clustering of latitude-longitude pairs.*

Resumo. *Este estudo analisa um dataset de hospedagens na plataforma Airbnb em Londres, onde diversas informações podem ser extraídas. Para isso, uma arquitetura com ferramentas de gerência de grandes volumes de dados foi aplicada, analisando desde n-gramas em nomes até clusterização de pares latitude-longitude.*

1. Introdução

Este relatório é um estudo sobre um dataset encontrado na plataforma *Kaggle* que possui informações relativas ao *Airbnb* em Londres. O estudo consiste em utilizar algumas ferramentas vistas em sala de aula para operações de leitura, manipulação e escrita em grandes volumes de dados. Nas seções seguintes, é descrito os objetivos do trabalho, a arquitetura de dados utilizadas, problemas com os resultados encontrados, como executar o projeto e uma conclusão.

2. Objetivos

O objetivo principal do trabalho foi aplicar os conhecimentos adquiridos durante o período em relação ao gerenciamento de grandes volumes de dados para realizar análises de dados em cima de um conjunto de dados escolhido, este com informações de *Airbnbs* da cidade de Londres em março de 2023[Wangondu 2023]. Esta análise é realizadada utilizando uma arquitetura apresentada na seção 3, com componentes como *Apache Spark* e *Apache Hadoop*.

Através de uma análise das colunas presentes no conjunto de dados, foram definidas algumas tarefas que poderiam ser realizadas para entender melhor o comportamento das hospedagens na capital inglesa. Estas tarefas envolviam diferentes tipos de processamento, utilizando consultas simples até processos como clusterização

A primeira tarefa definida tinha como objetivo analisar os nomes dos *airbnbs*, verificando unigramas e bigramas mais comuns em cada um deles. Outra tarefa buscava analisar a localização dos *hosts*, com a análise da quantidade de locais distintos e os locais de origem da maior parte deles. Ainda analisando a localização, definiu-se a busca por localizações onde os hosts eram cadastrados a mais tempo.

Por fim, duas tarefas foram definidas baseando-se no processo de clusterização. A primeira consistia em, utilizando latitude e longitude das hospedagens do conjunto de dados, realizar esta clusterização e definir algumas áreas principais. E, através destas áreas, definir, utilizando a coluna de preço, qual é o valor médio por hospedagem entre elas, e assim entender um comportamento das regiões londrinas no quesito financeiro.

3. Arquitetura

A arquitetura do projeto é constituída por algumas ferramentas vistas em sala de aula, das quais são *Hadoop File System*[1], *Hadoop MapReduce*[2], *Apache Spark*[3] e *Apache Hive*[4]. O fluxo de dados é representado pela figura 1 e pode ser descrita possuindo duas bases de dados principais em formatos *csv* (comma separated values) e *tsv* (tab-separated values). Estas bases são inseridas manualmente no HDFS para então haver o processamento pelo *Hadoop MapReduce* e o *Apache Spark*. Ao final do processo, os resultados são escritos em um diretório do *HDFS* para então ser consultado pelo *Apache Hive*.

O *HDFS*, como o próprio nome já diz, é um sistema de arquivos distribuídos que é altamente tolerante a falhas e projetado para ser executado em hardware de baixo custo. Ele é utilizado principalmente para armazenamento e leitura de grandes conjuntos de dados (em um modelo de write once, read many) e acesso de dados em streaming. No contexto do projeto, o *HDFS* é usado para o armazenamento do *data set* de entrada e dos dados processados.

Após os dados estarem armazenados em um sistema de arquivos próprio para *Big Data*, os mesmos são processados pelo *Hadoop MapReduce* e o *Apache Spark*. O *MapReduce* é um *framework* utilizado para processar grandes conjuntos de dados, em paralelo e tolerante a erros. Esta *framework* opera dividindo o conjunto de dados em subconjuntos independentes, onde cada um deles podem ser transformados (mapeados) paralelamente entre si. Com o resultado do mapeamento, acontecem as *tasks* de redução (*reduce*) para então os dados serem armazenados em um *file system*.

O *Spark*, assim como o *MapReduce*, também é utilizado para processamento de dados em grande escala. Basicamente é uma *engine* com esse propósito e possui uma *API* feita em diferentes linguagens de programação. Neste projeto, foi utilizado *pyspark* que é a *API* em *python*. O *Spark* também possui diversas instruções simplificadas para *Machine Learning*, processamento de gráficos, processamento de *streaming* e entre outras.

Com os dados processados e salvos no *HDFS*, é utilizado o *Apache Hive*, que tem como finalidade escrever, ler e gerenciar grandes volumes de dados em algum sistema de arquivos distribuídos utilizando *SQL* como sintaxe.

4. Resultados

As tarefas definidas no objetivo foram desenvolvidas utilizando a arquitetura da figura 1. As três primeiras tarefas utilizaram apenas estruturas de *MapReduce* do *Apache Hadoop* para seu processamento, enquanto as duas finais aplicaram funções do *Apache Spark*,

[1]https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[2]https://hadoop.apache.org/docs/r1.2.1/mapred$_{t}utorial.html$
[3]https://spark.apache.org/docs/3.4.1/
[4]https://cwiki.apache.org/confluence/display/Hive/

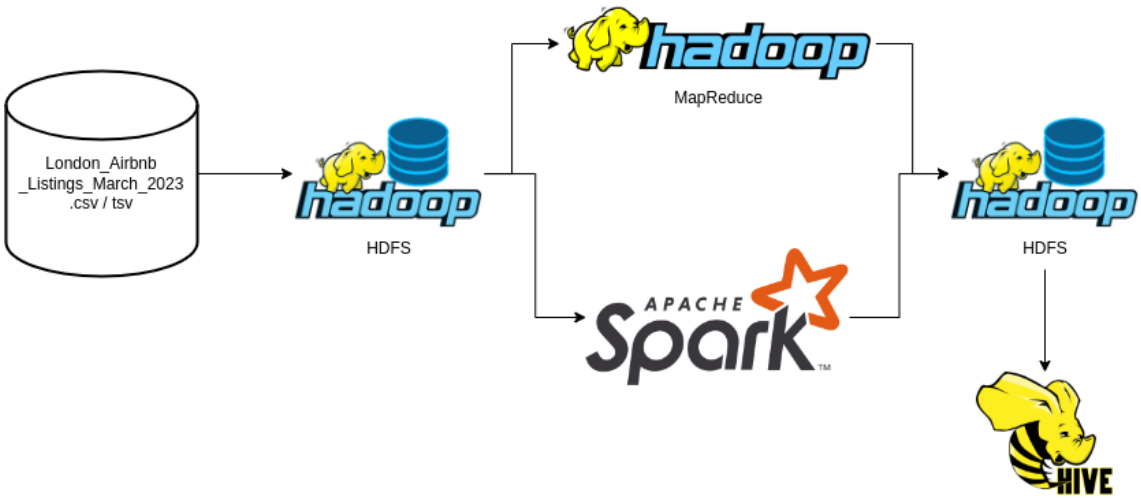

Figure 1. Arquitetura de Dados

implementado em *Python* para alcançar os processamentos desejados. Todas as tarefas tiveram seus objetivos alcançados com sucesso e terão seus resultados dispostos ao longo desta seção.

4.1. Nomes de airbnbs mais comuns - unigrama e bigrama

Para conseguir extrair dos dados as palavras e bigramas mais comuns em cada nome de hospedagem, utilizou-se de *MapReducers* que acessavam o HDFS e processavam, no *Mapper* os dados linha a linha, adicionando uma contagem para cada palavra, ou bigrama, que fosse encontrado. Esses dados tiveram uma etapa anterior, de pré-processamento, para remover caracteres especiais e deixar todos em letra minúscula. O *Reducer* então somava todos os valores com mesma chave e escrevia por fim, numa pasta de saída do HDFS, o conjunto de chave-valor contendo o n-grama e sua contagem.

Estes valores foram carregados para uma tabela dentro do *Apache Hive*, e consultados através de um comando `SELECT`, buscando os 15 maiores valores para cada chave de n-gram. Os resultados para as duas consultas estão nas tabelas 1 e 2

4.2. Localização de *hosts* distintos

A fim de analisar a localização de cada *host* das hospedagens londrinas, utilizou-se também uma arquitetura *MapReducer*, onde o *mapper* adicionava o valor um para cada valor da coluna *host_location*, e o *reducer* somava todos os valores para cada chave. Por fim foi retornado um *dataframe* com as localizações e as contagens de *hosts* cadastrados em cada lugar.

Esses dados foram disponibilizados em forma de tabela dentro do *hive*, e duas consultas foram realizadas. A primeira consulta utilizou um comando `COUNT`, para retornar a quantidade de valores distintos para a coluna, que teve retorno de **1558**, e uma consulta `SELECT`, com os 10 principais locais, exibidos na tabela 3

	Contagem
in	26482
flat	17154
room	16206
bedroom	14838
london	14374
apartment	10754
double	10049
2	9846
bed	9571
with	8932
1	7452
central	6007
house	5702
lovely	5691
spacious	5592

Table 1. Contagem de palavras

	Contagem
room in	6144
flat in	5803
double room	5460
central london	3868
bedroom flat	3235
2 bed	3134
apartment in	3048
1 bedroom	2848
2 bedroom	2754
double bedroom	2527
bedroom apartment	2463
1 bed	2446
bed flat	2429
bedroom in	2213
heart of	2139

Table 2. Contagem de bigramas

	Contagem
London, United Kingdom	42757
England, United Kingdom	10018
United Kingdom	664
Paris, France	227
New York, NY	128
Edinburgh, United Kingdom	103
Sydney, Australia	100
Richmond, United Kingdom	96
Kingston upon Thames, United Kingdom	87
Croydon, United Kingdom	87

Table 3. Dez principais *host_location*

4.3. Host locations que possuem hosts mais antigos

Seguindo uma análise em relação a localização do *host* de cada hospedagem, buscou-se analisar que locais possuiam *hosts* cadastrados a mais tempo, utilizando as colunas *host_location* e *host_since*. Neste caso, a arquitetura *MapReducer* funcionou de forma a percorrer cada coluna e retornar um par contendo as duas colunas. Estes pares eram então recebidos e escritos diretamente pelo *Reducer*.

Dentro de uma tabela criada com esses pares no *Hive*, realizou-se a consulta responsável por retornar o nome e a primeira data de cadastro dos cinco grupos de localização que possuiam as datas mais antigas, utilizando comandos SQL como `GROUP BY` e e `ORDER BY`. Estes locais estão dispostos na tabela 4, e, sendo a grande maioria de localização, teve Londres como local de onde surgiu o anfitrião mais antigo.

	Data
London, United Kingdom	2008-08-28
Los Angeles, CA	2009-03-21
Norwich, United Kingdom	2009-04-23
Berlin, Germany	2009-08-09
New York, NY	2009-09-22

Table 4. Cinco localizações com anfitriões mais antigos

4.4. Clusterizar latitude e longitude

A decisão de utilizar o PySpark para clusterização se deu por sua biblioteca de *Machine Learning*. O Algoritmo de clusterização é dividido em duas grandes etapas. A primeira é descobrir a quantidade ideal de clusters para, em seguida, os gerar com o algoritmo KMeans.

O principal parâmetro de entrada do *KMeans* é a quantidade de clusters em que as *features* devem ser separadas. Cada cluster tem uma centróide e a ”qualidade” da solução é definida pela soma dos erros quadrados entre as features e a sua centróide. Nesta tarefa, uma feature é constituída pelo par de Latitude e Longitude e o erro é a distância cartesiana entre as coordenadas do ponto e da centróide. É válido notar que, conforme o número de clusters se aproxima do número de features, a soma dos erros quadrados tende a zero.

Variando a quantidade de clusters e cruzando com a soma dos erros quadrados, foi gerado o gráfico que se encontra na figura 2, que possui uma curva chamada ”cotovelo”. O ”cotovelo” da curva possui o valor ideal de clusters dado que a partir disso, a variação da soma dos erros quadrados diminui drasticamente.

Por fim, o *KMeans* é aplicado com o valor encontrado de K igual a 8. Como resultado desta tarefa, foi gerado um arquivo parquet no hdfs contendo o cluster que cada coordenada pertence. Na tablela 5, temos a quantidade de pontos rotuladas por cada cluster. Esta tabela foi gerada a partir de uma consulta com o Hive.

Cluster	Quantidade
0	10198
1	5282
2	4768
3	13211
4	9589
5	3556
6	14124
7	14507

Table 5. Quantidade de coordenadas por cluster

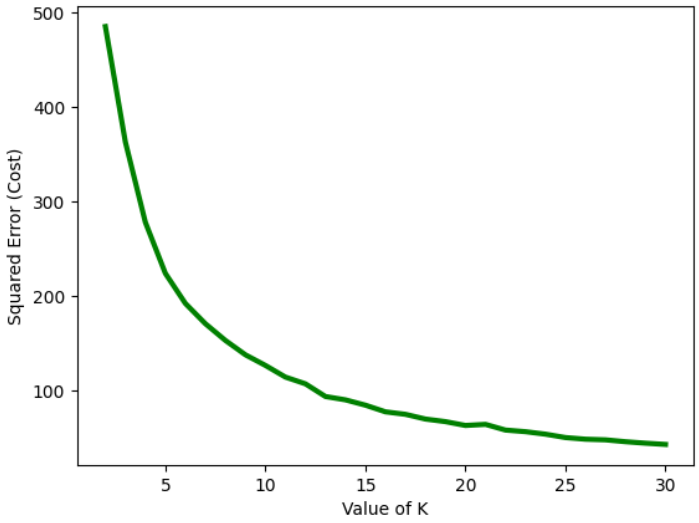

Figure 2. Curva Cotovelo

4.5. Preço Médio por cluster

Com os clusters gerados a partir da tarefa anterior e ainda utilizando o PySpark, é possível fazer uma simples análise de preço médio por cluster utilizando funções de agregação da API. A utilizada nesta tarefa foi o 'avg' e os resultados são encontrados na tabela 6

Cluster	Preço Médio
0	141.95
1	115.08
2	128.09
3	210.52
4	195.32
5	122.45
6	132.07
7	248.22

Table 6. Preço médio por cluster

5. Como Executar

Para realizar o processamento e consultar os dados gerados, primeiramente é necessário ter instalado na máquina o *Apache Spark*, *Apache Hadoop* e *Apache Hive*. Todos os códigos necessários para a execução se encontram em um repositório do Github [5]

Com as configurações prontas, o primeiro passo é a geração dos dados para consulta, que devem ser gerados seguindo os passos abaixo:

Passo 1: Criar um diretório /input no hdfs se ainda não existir, com o comando `hdfs dfs -mkdir /input`

[5]https://github.com/marizeraus/LondonAirbnbAnalysis

Passo 2: Adicionar os arquivos de input no hdfs executando os comandos `hdfs dfs -put input/London_Airbnb_Listings_March_2023.tsv /input` e `hdfs dfs -put input/London_Airbnb_Listings_March_2023.csv /input`

Passo 3: Compilar os arquivos Java e gerar as saídas do *MapReducer*, executando o script `run-all.sh`

Passo 4: Executar o arquivo em python, com o comando `python3 spark/run.py`

Após o final do processamento, todos os arquivos necessários se encontram salvos no *HDFS*, faltando apenas os seguintes passos para realizar a criação de tabelas e consulta de dados:

Passo 1: Para criar as tabelas e carregar os dados, acesse o *hive* e execute os comandos do arquivo `create-tables.md`

Passo 2: No arquivo `queries.md` estão dispostas as consultas necessárias para ver os dados de cada uma das tarefas especificadas, basta rodá-las no banco e verificar a saída.

6. Conclusão

Utilizando as ferramentas de gerenciamento de grandes volumes de dados aprendidas durante o período na disciplina, foi possível executar uma análise de um grande *dataset* de hospedagens na cidade de Londres cadastradas no *Airbnb*, com *Hadoop*, *Spark* e *Hive*, alcançando-se o objetivo deste trabalho.

Com base nos dados analisados, é possível perceber que, em relação ao nome das hospedagens, as palavras e bigramas mais frequentes são aquelas relacionadas a quarto e a cidade. As palavras **in**, **flat** e **room** se mostraram as mais citadas, e consequentemente os bigramas **room in** e **flat in**. Outras palavras que tiveram destaque foram ”bedroom” e ”london”, que também aparecem em bigramas relevantes.

Em relação a localização dos *hosts*, é possível concluir, sendo algo já esperado, que a grande maioria deles viria da cidade da hospedagem, Londres, estando apenas o país, Inglaterra, em segundo lugar do ranking. Apenas em quarto lugar, com uma porcentagem baixíssima de presença, aparece uma localização que não seja no local de origem, e este comportamento se repete na análise dos primeiros anfitriões cadastrados.

Na clusterização, 8 regiões foram definidas pelo algoritmo k-means, e com isso conseguimos separar estas regiões por preços, onde é possível concluir que a localização influencia o valor. Algumas regiões, como a número 7, tem preços bem mais elevado, enquanto algumas regiões, como a 1, possuem hospedagens que em média são mais baratas.

Assim, diversas análises foram possíveis de serem realizadas utilizando a arquitetura definida, com baixo tempo de processamento e resultados satisfátorios. O objetivo principal do trabalho foi concluído, e foi possível perceber grande aprendizado em relação as ferramentas estudadas ao longo do período.

References

Wangondu, M. (2023). London airbnb listings march 2023.