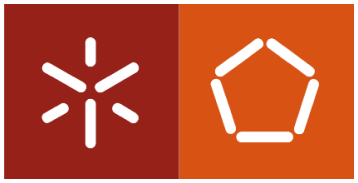

Universidade do Minho
Mestrado em Engenharia Informática
GGCD

Armazenamento e processamento de dados
Hadoop HDFS, Spark e Hive Metastore

Gonçalo Almeida - A84610
Pedro Ribeiro - PG42848
Raimundo Barros - PG42814

José Orlando Roque Nascimento Pereira

Mestrado em Engenharia Informática
2021

Abstract

O presente projeto consiste no desenvolvimento de métodos para analisar dados armazenados em ficheiros provenientes do *dataset* público da plataforma IMDb e armazená-los e processá-los num sistema distribuído apropriado denominado HDFS. Para tal, procedeu-se à utilização da linguagem de programação Java e da *framework* Spark para a análise e processamento de dados. Com base nos dados armazenados foi possível calcular várias métricas, como por exemplo: o género mais comum em cada década, o título mais bem classificado em cada ano e o top 10 atores que participaram em mais títulos diferentes. Para o armazenamento e processamento dos dados num sistema distribuído foi utilizado o Google Cloud ($PaaS$), visto que fornece toda a infraestrutura e serviços necessários para executar as tarefas propostas no trabalho. Numa fase de análise de resultados, observou-se que a solução implementada cumpriu a grande maioria dos objetivos propostos.

Keywords: HDFS, Spark, Google Cloud.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O presente relatório é o resultado da resolução do segundo trabalho prático da unidade curricular de Gestão de Grandes Conjuntos de Dados, do perfil de Ciência de Dados. O foco deste trabalho consiste na concretização e avaliação experimental de tarefas de armazenamento e processamento de dados. As tarefas serão implementadas com o *framework* Apache Spark, e o armazenamento e a execução das tarefas serão realizadas com o auxílio do sistema distribuído do Hadoop (HDFS) através do Google Cloud.

Os dados utilizados estão disponíveis no repositório público da IMDb (*name.basics.tsv.gz*, *title.basics.tsv.gz*, *title.principals.tsv.gz* e *title.ratings.tsv.gz*).

O objetivo será aplicar toda a computação necessária para processar os dados, de modo a dar resposta às tarefas propostas no trabalho.

Dessa forma, os ficheiros dos *datasets* serão carregados e, posteriormente, serão aplicadas operações com o Spark RDDs para processar os dados e devolver as informações requeridas.

No seguintes capítulos serão apresentadas as etapas desenvolvidas para a resolução das tarefas, serão também apresentadas as configurações realizadas para rodar o programa num ambiente distribuído, e as instruções que permitem executar o programa e realizar a leitura dos resultados.

Capítulo 2

Ferramentas

2.1 Hadoop HDFS

O Hadoop HFDS é utilizado para armazenar e analisar grandes quantidades de dados (estruturados e não estruturados). Num *cluster* Hadoop, os dados são armazenados e processados ao longo de diversos computadores de forma paralela.

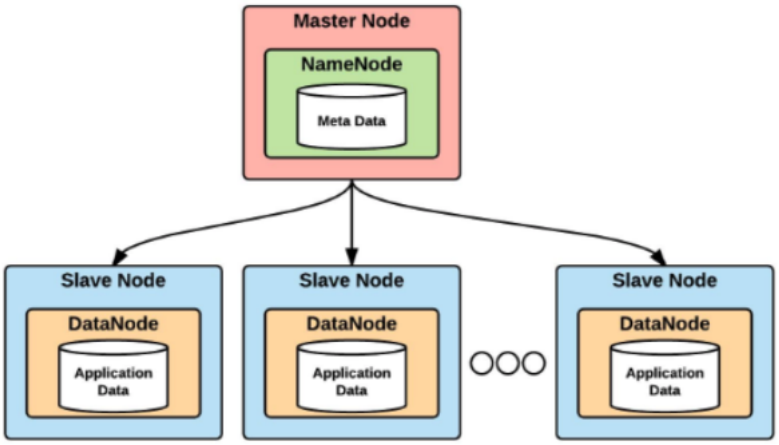

Figura 2.1: Arquitetura HDFS

Através de um JAR é possível fazer a leitura e o armazenamento de dados, além de realizar o processamento de dados designados através dos *jobs*. O processamento distribuído de dados, conforme mencionado no capítulo anterior, foi realizado através de operações com Spark RDDs. Para o armazenamento distribuído utilizamos o ambiente HDFS, onde o *NameNode* é um controlador principal (*master*) que mantém os metadados de todo o sistema de arquivos e permite os clientes conectarem-se ao servidor. Os dados, em geral, são armazenados nos *DataNodes*. No nosso projeto trabalhamos com duas arquiteturas, com 1 *datanode* na execução dos *jobs* numa única máquina local, e com 2 *datanodes* ao testar a solução na plataforma do Google Cloud.

2.2 Google Cloud (*PaaS*)

A criação do *cluster* é proveniente da Google Cloud Plataform, através do uso da ferramenta Dataproc, o que torna a criação de um *cluster* com o Apache Hadoop pré-instalado relativamente fácil. Para sua a criação de foram utilizadas as seguintes configurações descritas na tabela 2.2.

Nó	CPU Série	CPU Core	RAM	HDD
Master	E2	4	16GB	50GB
Worker		2	8GB	
Worker				

Após uma análise dos problemas, o grupo defende que os recursos escolhidos para o *cluster*, embora sejam modestos, são suficientes para as tarefas.

Para estabelecer uma conexão através do protocolo SSH foi utilizada a interface gráfica do manipulador de "instâncias de VM", onde este cria um separador no navegador que consta com PuTTY integrado, a alternativa seria a utilização do comando:

```
> gcloud compute ssh {MACHINE_NAME}
```

Após aceder ao *master* (*NameNode*) do *cluster* Hadoop, os ficheiros dos *datasets* devem ser importados para o ambiente *cloud*.

Listing 2.1: Template de importação de ficheiros

```
> cat {FICHEIRO} | gcloud compute ssh --zone={MACHINE_ZONE} \\
{MACHINE_NAME}-{NO} -- hdfs dfs -put - /{PATH}/{FICHEIRO}
```

Com os ficheiros necessários para correr o programa armazenados na cloud, a próxima etapa é executar o *job* através do ficheiro .jar e informar a classe que será executada. Importante também informar o nome do cluster e a região onde o mesmo se encontra.

Listing 2.2: Template de submissão de um *job*

```
> /bin/spark-submit --class {CLASS} --master local[8] {APP.JAR}
```

Capítulo 3

Resolução das Tarefas

Neste capítulo iremos abordar as nossas estratégias para a resolução das tarefas pedidas para este trabalho prático. Todas estas tarefas foram desenvolvidas com a ajuda de operações com Spark RDDs.

Antes de abordar as tarefas em causa, deve-se ter em consideração as seguintes notas:

- Para uma melhor compreensão das abordagens que tomamos, iremos representar os pares (tuplos) obtidos nas operações da seguinte forma (chave, valor);
- Para simplificar a descrição dos pares referenciados iremos utilizar as descrições dos atributos encontradas nos próprios conjuntos de dados, isto é, um par (id do título, id do ator) será representado por (*tconst*, *nconst*);
- Os conjuntos de dados públicos sobre os quais serão apresentados os resultados foram obtidos a 1 de Junho de 2021 da plataforma IMDb.

3.1 *Top Genres*

Esta tarefa consiste em determinar, para cada década, o género mais comum. Para tal, lê-mos o ficheiro *title.basics* e, ignorando as linhas com valores em falta para os atributos *startYear* e *genres*, associamos os géneros de cada título à década do próprio título, calculada através no ano de lançamento. Por fim, para cada década, realizamos a contagem das ocorrências de cada género e obtivemos aquele com o maior número de ocorrências, ou seja, o género mais comum, obtendo pares (década, género mais comum).

Na seguinte listagem são apresentados os resultados obtidos para esta tarefa.

Listing 3.1: Resultados da tarefa *Top Genres*

```
Decade of 1870-9: Short with 4 titles
Decade of 1880-9: Short with 55 titles
Decade of 1890-9: Short with 6041 titles
Decade of 1900-9: Short with 25476 titles
Decade of 1910-9: Short with 58857 titles
Decade of 1920-9: Short with 14020 titles
Decade of 1930-9: Short with 9304 titles
Decade of 1940-9: Short with 7735 titles
Decade of 1950-9: Drama with 41145 titles
Decade of 1960-9: Drama with 62462 titles
Decade of 1970-9: Drama with 101593 titles
Decade of 1980-9: Drama with 120472 titles
Decade of 1990-9: Drama with 195362 titles
Decade of 2000-9: Drama with 404749 titles
Decade of 2010-9: Drama with 777749 titles
Decade of 2020-9: Drama with 87473 titles
```

3.2 *Season Hits*

Esta tarefa consiste em determinar, para cada ano, o título mais bem classificado. Para tal, utilizamos vários Spark RDDs distintos.

Com o primeiro lê-mos o ficheiro *title.basics* e, ignorando as linhas com valores em falta para o atributo *startYear*, criamos pares (*tconst*, (*primaryTitle*, *startYear*)).

Com o segundo lê-mos o ficheiro *title.ratings* e criamos pares (*tconst*, *averageRating*).

De seguida, realizamos um *left join* do primeiro com o segundo e agrupamos os dados dos títulos pelo ano de lançamento. Por fim, para cada ano, calculamos o título com melhor classificação, obtendo pares (*startYear*, (*primaryTitle*, *averageRating*)).

Na seguinte listagem é apresentado um excerto dos resultados obtidos para esta tarefa.

Listing 3.2: Resultados da tarefa *Season Hits*

```
Year 2000: title 'Bimba' with a rating of 10.0
Year 2001: title 'Saving Sister Aimee' with a rating of 10.0
Year 2002: title 'Kasni rucak' with a rating of 10.0
Year 2003: title 'Republika Pescenica' with a rating of 10.0
Year 2004: title 'The Mark' with a rating of 10.0
Year 2005: title 'Birth Day Live!' with a rating of 10.0
Year 2006: title 'Second Nudist Turntable' with a rating of 10.0
Year 2007: title 'Episode #1.1' with a rating of 10.0
Year 2008: title 'Give Me Pink 3' with a rating of 10.0
Year 2009: title 'The Lady Masquerade' with a rating of 10.0
Year 2010: title 'Boicottatori di Nylon' with a rating of 10.0
```

3.3 *Top 10*

Esta tarefa consiste em determinar os 10 atores que participaram em mais títulos diferentes. Para tal, utilizamos vários Spark RDDs distintos.

Com o primeiro lê-mos o ficheiro *title.principals* e, apenas para os atores, calculamos o número de títulos e filtramos os 10 com um maior número de títulos, obtendo 10 pares (*nconst*, número de títulos).

Com o segundo lê-mos o ficheiro *name.basics* e, apenas para as linhas referentes aos 10 atores, obtivemos os respetivos nomes. No final associamos ambos os resultados, obtendo 10 pares (*primaryName*, número de títulos).

Na seguinte listagem são apresentados os resultados obtidos para esta tarefa.

Listing 3.3: Resultados da tarefa *Top 10*

```
Sameera Sherief with 9518 titles
Subhalekha Sudhakar with 6816 titles
Delhi Kumar with 6678 titles
Neha Gowda with 6055 titles
Sudha Chandran with 5685 titles
Jada Rowland with 5625 titles
Pallavi Ramisetty with 5606 titles
Manuela do Monte with 5200 titles
Peter Hobbs with 5037 titles
Giovanna Grigio with 5001 titles
```

3.4 *Base*

Esta tarefa consiste em determinar, para cada ator, o respetivo nome, idade, número de títulos em que participou, o intervalo de anos de atividade e a classificação média dos títulos em que participou. Para tal, utilizamos vários Spark RDDs distintos.

Com o primeiro lê-mos o ficheiro *title.basics* e, apenas para os atores, criamos pares (*tconst*, *nconst*).

Com o segundo lê-mos o ficheiro *title.ratings* e criamos pares (*tconst*, *averageRating*).

Com o terceiro realizamos um *left join* do primeiro com o segundo e calculamos, para cada ator, o número de títulos em que participou e a respetiva classificação média dos mesmos, obtendo pares (*nconst*, (número de títulos, classificação média)).

Com o quarto lê-mos o ficheiro *name.basics* e, apenas para os atores, calculamos as respetivas idades e criamos pares (*nconst*, (*primaryName*, idade)).

Por fim, realizamos um *left join* do primeiro com o quarto para agrupar todos os dados obtidos, obtendo pares (*nconst*, ((*primaryName*, idade), (número de títulos, classificação média)))).

Note-se que para esta tarefa não determinamos o intervalo de anos de atividade.

Na seguinte listagem é apresentado um excerto dos resultados obtidos para esta tarefa.

Listing 3.4: Resultados da tarefa *Base*

```
nm0001833:
  > name: Emily Watson
  > age: 54 years
  > number of titles: 60
  > mean rating from titles: 7.005

nm0358860:
  > name: Gwendoline Hamon
  > age: 51 years
  > number of titles: 38
  > mean rating from titles: 6.397368421052631
```

3.5 *Hits*

Esta tarefa consiste em determinar, para cada ator, os 10 títulos em que participou melhor classificados. Para tal, utilizamos vários Spark RDDs distintos.

Com o primeiro lê-mos o ficheiro *name.basics* e, apenas para os atores e para cada título em que participaram, criamos pares (*tconst*, *nconst*).

Com o segundo lê-mos o ficheiro *title.basics* e criamos pares (*tconst*, *primaryTitle*).

Com o terceiro lê-mos o ficheiro *title.ratings* e criamos pares (*tconst*, *averageRating*).

Com o quarto realizamos um *left join* do segundo com o terceiro, obtendo pares (*tconst*, (*primaryTitle*, *averageRating*)).

Por fim, realizamos um *left join* do primeiro com o quarto e, para cada ator, ordenamos os títulos em que participou pelo valor da classificação de forma decrescente e consideramos apenas os 10 primeiros títulos, obtendo pares (*nconst*, lista de pares (*primaryTitle*, *averageRating*)).

Na seguinte listagem é apresentado um excerto dos resultados obtidos para esta tarefa.

Listing 3.5: Resultados da tarefa *Hits*

```
nm0001833:
  > Please Remain Calm (average rating = 9.6)
  > Chernobyl (average rating = 9.4)
  > Einstein: Chapter Eight (average rating = 8.5)
  > Breaking the Waves (average rating = 7.8)
  > Episode #1.1 (average rating = 7.8)
  > Episode #1.2 (average rating = 7.8)
  > Meerkats (average rating = 7.8)
  > Episode #1.4 (average rating = 7.7)
  > Episode #1.2 (average rating = 7.6)
  > The Book Thief (average rating = 7.5)

nm0358860:
  > Tomb du ciel (average rating = 8.2)
  > Oil No More (average rating = 7.4)
  > Retour de flamme (average rating = 7.1)
  > Cassandre (average rating = 7.1)
  > Flics (average rating = 6.9)
  > Le pot de colle (average rating = 6.9)
  > Le loup gris (average rating = 6.9)
  > Une vie meilleure (average rating = 6.9)
  > Le saut de l'ange (average rating = 6.8)
  > Mort blanche (average rating = 6.8)
```

3.6 *Generation*

Esta tarefa consiste em determinar os nomes dos top 10 atores da mesma geração, em que consideramos número de títulos em que participaram como o termo de comparação. Para tal, utilizamos vários Spark RDDs distintos.

Com o primeiro lê-mos o ficheiro *title.principals* e, apenas para os atores, agrupamos os títulos em que participaram e realizamos a contagem dos mesmos, criando pares (*nconst*, número de títulos).

Com o segundo lê-mos o ficheiro *name.basics* e, ignorando as linhas com valores em falta para o atributo *startYear*, calculamos a década através no ano de lançamento e criamos pares (*nconst*, (*primaryName*, década)).

Por fim, realizamos um *left join* do primeiro com o segundo, agrupamos os dados pela década e determinamos, para cada década, os 10 atores com um maior número de títulos, obtendo pares (década, lista de pares (*primaryName*, número de títulos)).

Na seguinte listagem é apresentado um excerto dos resultados obtidos para esta tarefa.

Listing 3.6: Resultados da tarefa *Generations*

```
Decade of 2000-9:
  > Mutya Orquia with 675 titles
  > Sophia Valverde with 537 titles
  > Sienna Belle with 426 titles
  > Krystian Domagala with 416 titles
  > Maisa Silva with 407 titles
  > Addison Holley with 372 titles
  > Ji-young Kim with 347 titles
  > Millie Davis with 306 titles
  > Issac Ryan Brown with 280 titles
  > Francisca Aronsson with 259 titles

Decade of 2010-9:
  > Lorena Queiroz with 399 titles
  > Vitaliya Kornienko with 98 titles
  > Rin Furukawa with 54 titles
  > Kurumi Inagaki with 31 titles
  > Kue Lawrence with 26 titles
  > Gaius Lee DuPree with 18 titles
  > Tenta Banka with 18 titles
  > Maksim Boryak with 17 titles
  > Grumpy Cat with 14 titles
  > Anatoliy Naumov with 14 titles
```

3.7 *Friends*

Esta tarefa consiste em determinar, para cada ator, o respetivo conjunto de colaboradores, isto é, o conjunto dos outros atores que participaram nos mesmos títulos. Para tal, lê-mos o ficheiro *title.principals* e, apenas para os atores, agrupamos os respetivos ids pelos ids dos títulos. De seguida, para cada id do ator, calculamos os seus colaboradores no mesmo filme. Tendo, então, os colaboradores de cada ator para cada filme, juntamos os colaboradores de todos os filmes para o mesmo ator e removemos as ocorrências de colaboradores repetidas visto que dois atores podem ter colaborado em mais do que um filme. No final, obtivemos pares (*nconst*, conjunto de colaboradores).

Na seguinte listagem é apresentado um excerto dos resultados obtidos para esta tarefa.

Listing 3.7: Resultados da tarefa *Friends*

```
nm11999755:
  > nm11999755
  > nm12359978
  > nm12542823
  > nm11529410

nm11999812:
  > nm11999812
  > nm2809902
  > nm11997304
  > nm0000440
  > nm5684900
  > nm4724744
  > nm3472923
  > nm11999813
```

Capítulo 4

PySpark

Para termos de comparação entre duas formas de programação dentro da mesma ferramenta, o grupo decidiu explorar e implementar alguns exercícios no Pyspark. De notar que esta abordagem extra também serviu para teste dos resultados obtidos. O Pyspark é uma sub ferramenta do Apache Spark, no qual a principal especificidade é a utilização da linguagem Python, tendo como o seu núcleo um conjunto de ferramentas/noções como Spark SQL, DataFrames, Streaming, MLlib e Spark Core.

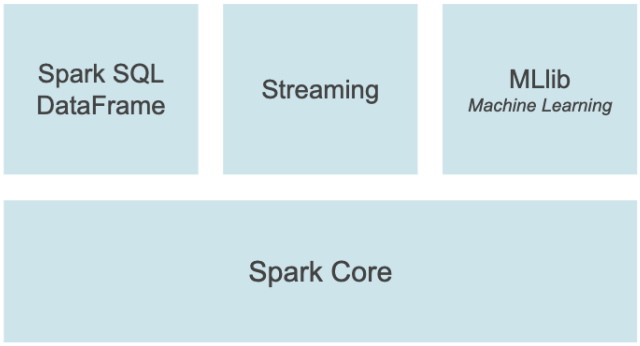

Figura 4.1: Arquitectura Pyspark

Para iniciar rapidamente o desenvolvimento no Pyspark, o grupo decidiu utilizar a ferramenta da Google Colaboratory, o código e o respectivo *notebook* que está disponível no repositório do Github do trabalho, dado pelo nome: *"PySpark.ipynb"* [3] [1].

4.1 Abordagem

Começou-se por fazer o *download* e a respetiva instalação do Spark e Hadoop e das respectivas dependências. Foi criada uma função de auxílio para proceder ao descarregamento e descompactamento dos ficheiros provenientes do IMDb.

Todas as métricas obtidas resultaram na utilização da classe Spark SQL e Dataframes, um conjunto de funções que podem ser executadas no mesmo tipo de conjunto de dados. [2]

Ambos os exercícios constam com um valor "Total"que indica qual foi o tempo de execução do código, que neste caso específico resume-se à utilização da função de auxílio, à leitura dos dados para Dataframes e às respectivas métricas.

4.1.1 *Top Genres*

Foram selecionadas apenas as colunas *'startYear'* e *'genres'*, posteriormente foi feita a separação pelo critério de separação ',' da última coluna criando uma nova linha para cada ocorrência. De forma a calcular as décadas, alteramos a primeira coluna e eliminamos o último caractere, obtendo assim, por exemplo: 1983 -> 198, tornando mais fácil a computação para cada década. Após agrupar os dados pela primeira coluna e de seguida pela segunda coluna foi calculado o valor a contagem dos acontecimentos. Para obter os resultados finais, executamos um agrupamento à primeira coluna e foram apenas escolhidos os valores máximos para cada década, obtendo o seguinte resultado 4.1.1.

Listing 4.1: Resultados da tarefa *Top Genres*

startYear	genero	Quantidade
187	Documentary	4
188	Sport	55
189	Sport	6041
190	Music	25476
191	Mystery	58857
192	Horror	14021
193	History	9304
194	Western	7735
195	Adult	41147
196	Documentary	62467
197	Documentary	101598
198	Adult	120630
199	Short	195214
200	Short	404800
201	Talk-Show	777901
202	Drama	87785

Total:

25.018404722213745

4.1.2 *Base*

Muito similar à resolução do último exercício, este apenas difere na utilização do join para juntar os 4 ficheiros de dados necessários para a resolução. O resultado final pode ser observado na seguinte lista 4.1.2

Listing 4.2: Resultados da tarefa *Base*

nconst	Nome	Idade	N Presen as	Na Atividade	Media de rating
nm0000086	Louis de Funes	69.0	59	70.0	7
nm0000198	Gary Oldman	63.0	120	37.0	7
nm0000354	Matt Damon	51.0	232	41.0	6
nm0000362	Danny DeVito	77.0	293	48.0	7
nm0000493	Jack Lemmon	76.0	146	66.0	7
nm0000767	Jean-Hugues Anglade	66.0	120	37.0	7
nm0000847	Theda Bara	70.0	27	98.0	6
nm0001079	Scatman Crothers	76.0	142	61.0	7
nm0001105	Linda Darnell	42.0	54	60.0	7
nm0002222	Christopher Bradley	60.0	13	27.0	7
nm0002941	Kevin Loader	65.0	28	27.0	7
nm0003283	Christopher Webb ...	0.0	1	0.0	6
nm0003678	Kate Reid	73.0	1	0.0	5
nm0003810	Colin Matthews	0.0	1	0.0	8
nm0003877	Dan Brown	51.0	2	8.0	8
nm0003926	Michael Huens	0.0	3	16.0	7
nm0004597	Adam Weissman	0.0	280	26.0	7
nm0004646	Antonio Sabato	78.0	31	45.0	6
nm0004709	Giuseppe Andrews	42.0	36	19.0	7
nm0004763	Mary J. Blige	50.0	151	39.0	6

only showing top 20 rows

Total:

148.80670166015625

4.1.3 *Hits*

A resolução deste exercício está centralizada na utilização de uma *Window*, uma função que através de uma coluna de exclusão consegue particionar e tratar cada grupo de dados de forma individual, desta forma é feita a partição segundo o id do ator e ordenar pela classificação dos respectivos títulos. A função *rank* finaliza o processo categorizando por classificações o resultado anterior. O resultado obtido pode ser observado na seguinte figura 4.1.3.

```
+---------+----------+--------------------+-------------+--------------+----+
|   nconst|    tconst|        primaryTitle|averageRating|   primaryName|rank|
+---------+----------+--------------------+-------------+--------------+----+
|nm0000086| tt3533882|Louis de Funès, d...|          8.6|Louis de Funès|   1|
|nm0000086| tt8768374|Le gendarme et l'...|          8.3|Louis de Funès|   2|
|nm0000086| tt0060474|La Grande Vadrouille|          7.9|Louis de Funès|   3|
|nm0000086| tt2949354|   Monsieur de Funès|          7.7|Louis de Funès|   4|
|nm0000086|tt12600988|La folle aventure...|          7.6|Louis de Funès|   5|
|nm0000086| tt0062083|               Oscar|          7.5|Louis de Funès|   6|
|nm0000086| tt0804908|Louis de Funès ou...|          7.5|Louis de Funès|   6|
|nm0000086| tt4307300|Louis de Funès et...|          7.5|Louis de Funès|   6|
|nm0000086| tt0069747|The Mad Adventure...|          7.4|Louis de Funès|   9|
|nm0000086| tt0049877|      Four Bags Full|          7.4|Louis de Funès|   9|
|nm0000086| tt0057967|          The Sucker|          7.4|Louis de Funès|   9|
|nm0000198| tt1432101|Take Flight: Gary...|          8.8|   Gary Oldman|   1|
|nm0000198| tt1266300|The Best Man for ...|          8.6|   Gary Oldman|   2|
|nm0000198| tt2395259|Jack White: Unstaged|          8.5|   Gary Oldman|   3|
|nm0000198| tt0110413|Léon: The Profess...|          8.5|   Gary Oldman|   3|
|nm0000198|tt10260592|Hollywood's Deadl...|          8.5|   Gary Oldman|   3|
|nm0000198| tt1345836|The Dark Knight R...|          8.4|   Gary Oldman|   6|
|nm0000198| tt5921272|   The Undead Series|          8.4|   Gary Oldman|   6|
|nm0000198| tt3038760|David Bowie: The ...|          8.3|   Gary Oldman|   8|
|nm0000198| tt7732340|Gary Oldman/Hugh ...|          8.3|   Gary Oldman|   8|
+---------+----------+--------------------+-------------+--------------+----+
only showing top 20 rows

Total:
----------------------------------------
164.04407143592834
----------------------------------------
```

Figura 4.2: PySpark: Resultado Hits

4.2 Análise de resultados

O *setup* do Google Colaboratory e a instalação Spark foram relativamente simples de realizar, a utilização de Python como linguagem de alto nível e a aproximação do PySpark Dataframes ao Pandas Dataframes torna mais fácil a sua implementação. O tempo de desenvolvimento, embora pequeno, tornou-se justificável visto que neste momento pode-se comparar resultados nas duas abordagens. O tempo de execução de cada atividade foi melhor do que o grupo esperava, visto que a plataforma de desenvolvimento faz a gestão de recursos de acordo com o número de utilizadores. Desta forma, sendo o ambiente de desenvolvimento significativamente inferior ao do Java Spark (Google Cloud), ambos tiveram resultados muito similares no tempo de execução.

Capítulo 5

Conclusões

O objetivo deste trabalho prático consistiu no aumento da experiência no âmbito do desenvolvimento de aplicações para trabalhar com grandes quantidades de dados utilizando diversos ficheiros. Além disso, o projeto visava a consolidação da utilização de processos de processamento de dados com as ferramentas Spark num ambiente distribuído HDFS, realçando a utilidade destas ferramentas para a resolução de problemas. Como trabalho futuro, seria interessante abordar estes problemas, não só através de operações com Spark RDDs, mas também com SQL. De qualquer forma, os resultados obtidos foram satisfatórios e consideramos que cumprimos a grande parte dos objetivos propostos.

Apêndice A

Resultados das Tarefas em HDFS

As imagens presentes neste **Apêndice A** são respectivas às execuções das abordagens realizadas para a resolver os problemas, executado no cluster HDFS da Google Cloud Platform.

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit   --class TopGenres   --master local[8]   App-1.0-SNAPSHOT.jar
21/06/03 01:24:14 INFO org.sparkproject.jetty.util.log: Logging initialized @2377ms to org.sparkproject.jetty.uti
21/06/03 01:24:14 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2021-01-14T16:44:28.
c6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/03 01:24:14 INFO org.sparkproject.jetty.server.Server: Started @2470ms
21/06/03 01:24:14 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@7a34b7b8{HTTP/1.
0.0.0.0:42473}
21/06/03 01:24:17 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 01:24:17 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/03 01:24:17 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/03 01:24:18 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded & initialized native-
21/06/03 01:24:18 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
Decade of 1870-9: Short with 4 titles
Decade of 1880-9: Short with 55 titles
Decade of 1890-9: Short with 6041 titles
Decade of 1900-9: Short with 25476 titles
Decade of 1910-9: Short with 58857 titles
Decade of 1920-9: Short with 14021 titles
Decade of 1930-9: Short with 9304 titles
Decade of 1940-9: Short with 7735 titles
Decade of 1950-9: Drama with 41147 titles
Decade of 1960-9: Drama with 62467 titles
Decade of 1970-9: Drama with 101598 titles
Decade of 1980-9: Drama with 120630 titles
Decade of 1990-9: Drama with 195214 titles
Decade of 2000-9: Drama with 404800 titles
Decade of 2010-9: Drama with 777901 titles
Decade of 2020-9: Drama with 87785 titles
21/06/03 01:24:51 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@7a34b7b8{HTTP/1.1, (http/1.
```

Figura A.1: Resultados da tarefa *Top Genres*

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit   --class SeasonHits  --master local[8]   App-1.0-SNAPSHOT.jar

21/06/03 01:28:49 INFO org.sparkproject.jetty.util.log: Logging initialized @2346ms to org.sparkproject.jetty.ut
il.log.Slf4jLog
21/06/03 01:28:49 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2021-01-14T16:44:28.
689Z; git: 238ec6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/03 01:28:49 INFO org.sparkproject.jetty.server.Server: Started @2447ms
21/06/03 01:28:49 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@7a34b7b8{HTTP/1.
1, (http/1.1)}{0.0.0.0:37377}
21/06/03 01:28:53 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 01:28:53 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/03 01:28:53 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/03 01:28:53 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 01:28:53 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #1,5,main])
interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThreadPoolExecutor.java
:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
21/06/03 01:28:54 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded & initialized native-
zlib library
21/06/03 01:28:54 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/03 01:28:54 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
Year 1874: title 'Passage de Venus' with a rating of 6.9
Year 1877: title 'La Rosace Magique' with a rating of 7.1
Year 1878: title 'Sallie Gardner at a Gallop' with a rating of 7.4
Year 2015: title 'Shade of Music' with a rating of 10.0
Year 2016: title 'YanYuan' with a rating of 10.0
Year 2017: title 'The Impossible Joy' with a rating of 10.0
Year 2018: title 'Beyond Repair (Pilot)' with a rating of 10.0
Year 2019: title 'Danny Gonzalez: Help Let Me Go' with a rating of 10.0
Year 2020: title 'Episode VII: Streamers of the Lost Art (of Conversation)' with a rating of 10.0
Year 2021: title 'Fidèle' with a rating of 10.0
21/06/03 01:30:40 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@7a34b7b8{HTTP/1.1, (http/1
.1)}{0.0.0.0:0}
gamesredbull9@cluster-3e33-m:~$
```

Figura A.2: Resultados da tarefa *Seanson Hits*

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit   --class Top10   --master local[8]   App-1.0-SNAPSHOT.jar
21/06/03 01:25:56 INFO org.sparkproject.jetty.util.log: Logging initialized @2431ms to org.sparkproject.jetty.ut
il.log.Slf4jLog
21/06/03 01:25:56 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2021-01-14T16:44:28.
689Z; git: 238ec6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/03 01:25:56 INFO org.sparkproject.jetty.server.Server: Started @2525ms
21/06/03 01:25:56 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@7a34b7b8{HTTP/1.
1, (http/1.1)}{0.0.0.0:32975}
21/06/03 01:25:59 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 01:25:59 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/03 01:25:59 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/03 01:26:00 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded & initialized native-
zlib library
21/06/03 01:26:00 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/03 01:27:07 INFO org.apache.hadoop.mapred.FileInputForm
at: Total input files to process : 1
21/06/03 01:27:07 INFO org.apache.hadoop.net.NetworkTopology:
 Adding a new node: /default-rack/10.186.0.4:9866
21/06/03 01:27:07 INFO org.apache.hadoop.net.NetworkTopology:
 Adding a new node: /default-rack/10.186.0.2:9866
Sameera Sherief with 9518 titles
Subhalekha Sudhakar with 6816 titles
Delhi Kumar with 6678 titles
Neha Gowda with 6055 titles
Sudha Chandran with 5685 titles
Jada Rowland with 5625 titles
Pallavi Ramisetty with 5606 titles
Manuela do Monte with 5200 titles
Peter Hobbs with 5037 titles
Giovanna Grigio with 5001 titles
21/06/03 01:27:21 INFO org.sparkproject.jetty.server.Abstract
Connector: Stopped Spark@7a34b7b8{HTTP/1.1, (http/1.1)}{0.0.0
.0:0}

Time: 86447 ms
```

Figura A.3: Resultados da tarefa *Top 10*

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit --class Base --master local[8] App-1.0-SNAPSHOT.jar
21/06/03 14:32:35 INFO org.sparkproject.jetty.util.log: Logging initialized @2754ms to org.sparkproject.jetty.ut
il.log.Slf4jLog
21/06/03 14:32:36 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2021-01-14T16:44:28.
689Z; git: 238ec6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/03 14:32:36 INFO org.sparkproject.jetty.server.Server: Started @2868ms
21/06/03 14:32:36 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@b0a5b10{HTTP/1.1
, (http/1.1)}{0.0.0.0:42291}
21/06/03 14:32:40 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 14:32:40 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/03 14:32:40 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/03 14:32:40 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 14:32:40 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/03 14:32:40 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #1,5,main])
interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThreadPoolExecutor.java
:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
21/06/03 14:32:40 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/03 14:32:40 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/03 14:32:41 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded & initialized native-
zlib library
21/06/03 14:32:41 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/03 14:32:41 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/03 14:32:41 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
nm0272284:
  > name: Andrzej Ferenc
  > age: 62 years
  > number of titles: 1
  > mean rating from titles: 6.4
nm0696614:
  > name: Uwe Preuss
  > age: 60 years
  > number of titles: 50
  > mean rating from titles: 6.345999999999999
nm2957371:
  > name: Genevieve Sibayan
  > age: 38 years
  > number of titles: 4
  > mean rating from titles: 6.475
nm3300509:
  > name: Selam Tadese
  > age: 41 years
  > number of titles: 2
  > mean rating from titles: 7.2
nm0708542:
  > name: Alby Ramos
  > age: 70 years
  > number of titles: 1
  > mean rating from titles: 7.4
nm0446987:
  > name: Paul Kelman
  > age: 54 years
  > number of titles: 5
  > mean rating from titles: 6.859999999999999
nm1967053:
  > name: AmaLee
  > age: 29 years
  > number of titles: 169
  > mean rating from titles: 7.576923076923077
21/06/03 14:35:25 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@b0a5b10{HTTP/1.1, (http/1.
1)}{0.0.0.0:0}
gamesredbull9@cluster-3e33-m:~$
```

Figura A.4: Resultados da tarefa *Base*

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit  --class Hits --master local[8] App-1.0-SNAPSHOT.jar
21/06/04 08:37:00 INFO org.sparkproject.jetty.util.log: Logging initialized @2413ms to org.sparkproject.jetty.ut
il.log.Slf4jLog
21/06/04 08:37:00 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2021-01-14T16:44:28.
689Z; git: 238ec6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/04 08:37:00 INFO org.sparkproject.jetty.server.Server: Started @2511ms
21/06/04 08:37:00 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@286e0788{HTTP/1.
1, (http/1.1)}{0.0.0.0:41127}
21/06/04 08:37:04 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/04 08:37:04 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/04 08:37:04 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/04 08:37:04 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/04 08:37:04 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process : 1
21/06/04 08:37:04 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/04 08:37:04 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/04 08:37:05 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded & initialized native-
zlib library
21/06/04 08:37:05 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/04 08:37:05 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/04 08:37:05 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]


nm7342820:
  > As I Am Suffering from Kadhal (average rating = 7.8)
  > Suffering from chronic fighting (average rating = 7.2)
  > Meet the sufferers (average rating = 7.0)
  > Demonte Colony (average rating = 7.0)
  > Suffering from perfection deficiency (average rating = 7.0)
  > Jil Jung Juk (average rating = 6.8)
  > Suffering from high expectations (average rating = 6.6)
  > Suffering from external problems (average rating = 6.4)

nm0490224:
  > The Naked and the Wicked (average rating = 8.4)
  > The Naples Beat (average rating = 7.9)
  > Flowers from Alexander (average rating = 7.8)
  > 13 Rue Madeleine (average rating = 6.9)
  > Una donna ha ucciso (average rating = 6.9)
  > Terrore sulla città (average rating = 6.7)
  > Yvonne of the Night (average rating = 6.5)
  > Sul ponte dei sospiri (average rating = 6.5)
  > The Sergeant (average rating = 6.5)
  > Black Magic (average rating = 6.4)

nm6619439:
  > Our Company 2011-2016 (average rating = 9.6)
  > Untitled Powtoon for the Competition (average rating = 9.5)
  > The Hoppus Leo (average rating = 9.5)
  > End This Season!! (average rating = 8.9)
  > Untitled Powtoon for Anybody (average rating = 8.9)
  > The Twenty-Eight Hits for Laughs (average rating = 6.6)
```

Figura A.5: Resultados da tarefa *Hits*

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit --class Generation --master local[8] App-1.0-SNAPSHOT.jar
21/06/04 09:03:17 INFO org.sparkproject.jetty.util.log: Logging initialized @2464ms to org.sparkproject.jetty.ut
il.log.Slf4jLog
21/06/04 09:03:17 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2021-01-14T16:44:28.
689Z; git: 238ec6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/04 09:03:17 INFO org.sparkproject.jetty.server.Server: Started @2583ms
21/06/04 09:03:17 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@7a34b7b8{HTTP/1.
1, (http/1.1)}{0.0.0.0:36817}
21/06/04 09:03:21 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #0,5,main])
interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThreadPoolExecutor.java
:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
21/06/04 09:03:21 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.2:9866
21/06/04 09:03:21 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rack/10.186.0.4:9866
21/06/04 09:03:22 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded & initialized native-
zlib library
21/06/04 09:03:22 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
21/06/04 09:03:22 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.gz]
Decade of 2000-9:
  > Mutya Orquia with 675 titles
  > Sophia Valverde with 537 titles
  > Sienna Belle with 426 titles
  > Krystian Domagala with 416 titles
  > Maisa Silva with 407 titles
  > Addison Holley with 372 titles
  > Ji-young Kim with 347 titles
  > Millie Davis with 306 titles
  > Issac Ryan Brown with 280 titles
  > Francisca Aronsson with 259 titles

Decade of 2010-9:
  > Lorena Queiroz with 399 titles
  > Vitaliya Kornienko with 97 titles
  > Rin Furukawa with 54 titles
  > Kurumi Inagaki with 31 titles
  > Kue Lawrence with 25 titles
  > Tenta Banka with 18 titles
  > Maksim Boryak with 17 titles
  > Gaius Lee DuPree with 17 titles
  > Grumpy Cat with 14 titles
  > Anatoliy Naumov with 14 titles

Decade of 2020-9:
  > Lilah Sykes with 10 titles
  > Noah Lira with 2 titles
21/06/04 09:04:44 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@7a34b7b8{HTTP/1.1, (http/1
.1)}{0.0.0.0:0}
```

Figura A.6: Resultados da tarefa *Generation*

```
gamesredbull9@cluster-3e33-m:~$ /bin/spark-submit --class Friends --master local[8] App-1.0-S
NAPSHOT.jar
21/06/03 13:35:23 INFO org.sparkproject.jetty.util.log: Logging initialized @2872ms to org.sp
arkproject.jetty.util.log.Slf4jLog
21/06/03 13:35:23 INFO org.sparkproject.jetty.server.Server: jetty-9.4.36.v20210114; built: 2
021-01-14T16:44:28.689Z; git: 238ec6997c7806b055319a6d11f8ae7564adc0de; jvm 1.8.0_292-b10
21/06/03 13:35:23 INFO org.sparkproject.jetty.server.Server: Started @2986ms
21/06/03 13:35:23 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnect
or@7a34b7b8{HTTP/1.1, (http/1.1)}{0.0.0.0:33375}
21/06/03 13:35:27 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Thread[GetFi
leInfo #1,5,main]) interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java
:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(Exec
utorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThre
adPoolExecutor.java:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
21/06/03 13:35:27 INFO org.apache.hadoop.mapred.FileInputFormat: Total input files to process
 : 1
21/06/03 13:35:27 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rac
k/10.186.0.4:9866
21/06/03 13:35:27 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default-rac
k/10.186.0.2:9866
21/06/03 13:35:28 INFO org.apache.hadoop.io.compress.zlib.ZlibFactory: Successfully loaded &
initialized native-zlib library
21/06/03 13:35:28 INFO org.apache.hadoop.io.compress.CodecPool: Got brand-new decompressor [.
gz]
nm0299998:
  > nm0299998
  > nm0960379
nm9254657:
  > nm1546474
  > nm9254657
  > nm5931756
nm5817847:
  > nm6561765
  > nm5817847
  > nm8077090
nm5601382:
  > nm3354170
  > nm5434991
  > nm5601382
nm12135620:
  > nm12135623
  > nm12135620
  > nm12135624
nm5307140:
  > nm6861641
  > nm5307140
```

Figura A.7: Resultados da tarefa *Friends*

Bibliografia

[1] Repositório: Trabalho
https://github.com/PedromfRibeiro/GGCD/tree/main/Projeto2

[2] Module Context of PySpark
https://spark.apache.org/docs/2.3.1/api/python/pyspark.sql.html

[3] Repositório: PySpark Notebook
https://github.com/PedromfRibeiro/GGCD/blob/main/PySpark.ipynb