

Gestão de Grandes Conjuntos de Dados

1º Trabalho Prático

-

Mestrado em Engenharia Informática
Universidade do Minho

Grupo nº 8

PG41080	João Ribeiro Imperadeiro
PG41081	José Alberto Martins Boticas
PG41091	Nelson José Dias Teixeira
PG41851	Rui Miguel da Costa Meira

21 de Abril de 2020

Conteúdo

1	Introdução	3
2	Implementação	4
2.1	Arranque do <i>cluster</i>	4
2.2	1ª Tarefa	5
2.2.1	Criação da tabela <i>HBase</i>	5
2.2.1.1	Alternativa	6
2.2.2	Transferência do ficheiro para a plataforma <i>Hadoop HDFS</i>	6
2.2.2.1	1ª Alternativa	7
2.2.2.2	2ª Alternativa	7
2.2.3	População da tabela <i>HBase</i>	7
2.3	2ª Tarefa	8
2.3.1	Criação da tabela <i>HBase</i>	8
2.3.1.1	Alternativa	9
2.3.2	Transferência de ficheiros para a plataforma <i>Hadoop HDFS</i>	9
2.3.2.1	1ª Alternativa	9
2.3.2.2	2ª Alternativa	10
2.3.3	População da tabela <i>HBase</i>	10
2.3.3.1	Nome, datas de nascimento e morte do ator	10
2.3.3.2	Número total de filmes + Top 3	11
3	Conclusão	13
A	Observações	14

Lista de Figuras

2.1	Arranque do <i>cluster</i>	4
2.2	1ª Tarefa - Conversão do ficheiro <i>"title.basics.tsv.gz"</i> para o formato <i>.tsv</i>	5
2.3	1ª Tarefa - <i>Dockerfile</i>	5
2.4	1ª Tarefa - <i>Dockerfile</i> - Opções de execução	5
2.5	1ª Tarefa - Modelo da tabela <i>HBase "movies"</i>	6
2.6	1ª Tarefa : Alternativa - Acesso à <i>HBase shell</i>	6
2.7	1ª Tarefa : Alternativa - Criação da tabela <i>HBase "movies"</i>	6
2.8	1ª Tarefa : Alternativa - Remoção da tabela <i>HBase "movies"</i>	6
2.9	1ª Tarefa - Criação da pasta data na plataforma <i>Hadoop HDFS</i>	6
2.10	1ª Tarefa : 1ª Alternativa - Transferência do ficheiro <i>"title.basics.tsv"</i> para a plataforma <i>Hadoop HDFS</i>	7
2.11	1ª Tarefa : 2ª Alternativa - Transferência do ficheiro <i>"title.basics.tsv"</i> para a plataforma <i>Hadoop HDFS</i>	7
2.12	1ª Tarefa - Esquema do paradigma <i>MapReduce</i>	7
2.13	2ª Tarefa - Modelo da tabela <i>HBase "actors"</i>	8
2.14	2ª Tarefa : Alternativa - Criação da tabela <i>HBase "actors"</i>	9
2.15	2ª Tarefa : Alternativa - Remoção da tabela <i>HBase "actors"</i>	9
2.16	2ª Tarefa : 1ª Alternativa - Transferência dos 3 ficheiros de <i>input</i> para a plataforma <i>Hadoop HDFS</i>	9
2.17	2ª Tarefa : 2ª Alternativa - Transferência dos 3 ficheiros de <i>input</i> para a plataforma <i>Hadoop HDFS</i>	10
2.18	2ª Tarefa - <i>Actor2Details</i> - Esquema do paradigma <i>MapReduce</i>	11
2.19	2ª Tarefa - <i>Actor2Movies</i> - 1º Esquema do paradigma <i>MapReduce</i>	12
2.20	2ª Tarefa - <i>Actor2Movies</i> - 2º Esquema do paradigma <i>MapReduce</i>	12

Capítulo 1

Introdução

Na primeira parte deste trabalho prático é requerida a concretização e avaliação experimental de tarefas de armazenamento e processamento de dados utilizando as ferramentas computacionais *Hadoop HDFS*, *HBase* e, ainda, o paradigma *MapReduce*. Por forma a realizar estas tarefas, os dados a utilizar para tal efeito correspondem ao conjunto de dados público do *IMDB*, que se encontram disponíveis em:

<https://www.imdb.com/interfaces/>

Ao longo deste documento vão também ser expostos todos os passos tomados durante a implementação das tarefas pedidas neste projeto, incluindo as decisões tomadas pelos elementos deste grupo a nível de algoritmos e parâmetros de configuração. Para além disso são ainda apresentadas todas as instruções que permitem executar e utilizar corretamente os programas desenvolvidos. Por fim, na fase final deste manuscrito, são exibidos os objetivos atingidos após a realização das tarefas propostas.

De salientar ainda que durante os capítulos que se seguem são identificadas algumas alternativas para concretizar as tarefas indicadas neste trabalho prático.

Capítulo 2

Implementação

Tal como foi enunciado anteriormente, neste projeto é globalmente solicitada a elaboração de duas tarefas. Apresentam-se de seguida as mesmas:

1. Carregar os dados do ficheiro *"title.basics.tsv.gz"* para uma tabela *HBase*;
2. Utilizando a tabela *HBase* do ponto acima e os restantes ficheiros presentes no *dataset* mencionado no capítulo anterior, computar os dados necessários para apresentar para cada ator uma página. Esta última deve conter:
 - nome, datas de nascimento e morte;
 - número total de filmes em que participou como ator;
 - títulos dos três filmes com melhor cotação em que participou.

Estes dados devem ser armazenados numa tabela *HBase*.

Nas próximas secções são evidenciadas as implementações para cada uma destas tarefas bem como algumas sugestões alternativas que poderiam ser tomadas em consideração.

2.1 Arranque do *cluster*

À semelhança do que foi realizado no guião nº 4 desta unidade curricular, os elementos deste grupo utilizaram a plataforma *Hadoop deployment* com recurso à ferramenta *docker-compose*. Este utensílio computacional encontra-se disponível em:

<https://github.com/big-data-europe/docker-hbase>

De forma a proceder ao arranque do *cluster* contido neste repositório basta simplesmente invocar a seguinte instrução:

```
docker-compose -f docker-hbase/docker-compose-distributed-local.yml up
```

Figura 2.1: Arranque do *cluster*

Após a execução desta instrução, o *cluster* que irá integrar este projeto encontra-se corretamente instanciado e, como tal, pode-se proceder à realização das tarefas mencionadas no topo desta página.

2.2 1ª Tarefa

Após descarregar o ficheiro *"title.basics.tsv.gz"* presente na hiperligação do capítulo anterior, os elementos que compõem este grupo optaram por converter o mesmo no formato *.tsv*. A tomada desta decisão deve-se ao facto deste último permitir a partição de dados (isto é, potencia o **paralelismo**), ao contrário do formato *.gz* (*gzip*), e, ainda, ser mais rápido e eficiente no processo de descompressão quando comparado com o formato *.bz2* (*bzip2*). Mostra-se na seguinte figura a instrução associada à descompressão do ficheiro *"title.basics.tsv.gz"*:

```
gzip -d title.basics.tsv.gz
```

Figura 2.2: 1ª Tarefa - Conversão do ficheiro *"title.basics.tsv.gz"* para o formato *.tsv*

Antes de observar os passos relativos à realização desta tarefa, passos esses que se encontram explicitamente indicados nos próximos subcapítulos, é importante salientar que a execução das soluções elaboradas nas secções 2.1.1 e 2.1.3 são efetuadas com recurso a um ficheiro denominado por *Dockerfile*. De forma a entender melhor a configuração do mesmo, revela-se a seguir o seu conteúdo:

```
FROM bde2020/hadoop-base
COPY target/TP1-1.0-SNAPSHOT.jar /
ENTRYPOINT ["hadoop", "jar", "/TP1-1.0-SNAPSHOT.jar", "ClassName"]
```

Figura 2.3: 1ª Tarefa - *Dockerfile*

Após esta observação, indica-se ainda as opções adotadas para a execução do ficheiro *Dockerfile* com o intuito de garantir uma execução válida das soluções implementadas:

```
--network docker-hbase_default
--env-file ../docker-hbase/hadoop.env
--env-file ../docker-hbase/hbase-distributed-local.env
```

Figura 2.4: 1ª Tarefa - *Dockerfile* - Opções de execução

2.2.1 Criação da tabela *HBase*

De forma a criar a tabela *HBase* intrínseca a esta tarefa, foi implementada uma classe *Java*, ***CreateTableMovies***, que, após conectar-se com a base de dados não relacional *HBase*, trata da sua criação e configuração. Durante esse processo, é produzida apenas uma família de colunas, intitulada por ***details***, onde será armazenada toda a informação associada aos dados do ficheiro *"title.basics.tsv.gz"*.

De notar também que atribuiu-se o nome ***movies*** à tabela gerada, tal como o nome da classe *Java* transparece.

Foi também criada uma classe *Java* adicional, ***DeleteTableMovies***, que trata de eliminar a tabela descrita anteriormente. Esta foi desenvolvida com o intuito de remover a tabela em causa caso esta deixe de ser necessária no futuro.

Apresenta-se de seguida o modelo da tabela *HBase* pretendido para a concretização desta tarefa:

	columnFamily			details				
	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
idMovie1
...
idMovieN

Figura 2.5: 1ª Tarefa - Modelo da tabela *HBase "movies"*

2.2.1.1 Alternativa

Uma possibilidade válida para realizar todo o processo associado à criação da tabela requerida seria utilizar a *HBase shell* de forma direta. Exibe-se de seguida as respetivas instruções:

```
docker run -it
--network docker-hbase_default
--env-file docker-hbase/hbase-distributed-local.env
bde2020/hbase-base hbase shell
```

Figura 2.6: 1ª Tarefa : Alternativa - Acesso à *HBase shell*

```
hbase(main):001:0> create "movies", "details"
```

Figura 2.7: 1ª Tarefa : Alternativa - Criação da tabela *HBase "movies"*

Quanto à remoção da mesma tabela, à semelhança do procedimento tomado para a sua criação, adota-se a estratégia de usufruir explicitamente o mecanismo disponibilizado pela *HBase shell*:

```
hbase(main):001:0> disable "movies"
hbase(main):002:0> drop "movies"
```

Figura 2.8: 1ª Tarefa : Alternativa - Remoção da tabela *HBase "movies"*

2.2.2 Transferência do ficheiro para a plataforma *Hadoop HDFS*

De maneira a proceder ao carregamento do ficheiro *"title.basics.tsv"* para a plataforma *Hadoop HDFS* existem duas possibilidades. Antes de exibir estas últimas alternativas, foi criada uma pasta na plataforma *Hadoop HDFS*, denominada por *data*, onde serão colocados todos os ficheiros de *input* necessários. Exibe-se de seguida a instrução para tal efeito:

```
docker run --network docker-hbase_default
--env-file docker-hbase/hadoop.env
bde2020/hadoop-base hdfs dfs -mkdir /data
```

Figura 2.9: 1ª Tarefa - Criação da pasta *data* na plataforma *Hadoop HDFS*

Após a exposição deste comando, destacam-se nos próximos subcapítulos as duas alternativas mencionadas acima.

2.2.2.1 1ª Alternativa

Nesta possibilidade evidencia-se o campo **source** que corresponde à diretoria da pasta que contém o ficheiro *"title.basics.tsv"*. Dito isto, apresenta-se agora a primeira alternativa:

```
docker run --network docker-hbase_default
--env-file docker-hbase/hadoop.env
--mount type=bind,source="/path/to/local/folder/data",target=/data
bde2020/hadoop-base hdfs dfs -put /data/title.basics.tsv /data
```

Figura 2.10: 1ª Tarefa : 1ª Alternativa - Transferência do ficheiro *"title.basics.tsv"* para a plataforma *Hadoop HDFS*

2.2.2.2 2ª Alternativa

Esta opção corresponde ao modo interativo de execução disponibilizado pela instrução *docker run*. Uma vez feita esta observação, expõe-se a seguir a segunda alternativa:

```
docker run -it
--network docker-hbase_default
--env-file docker-hbase/hadoop.env
bde2020/hadoop-base bash

curl https://datasets.imdbws.com/title.basics.tsv.gz | gunzip |
hdfs dfs -put - hdfs://namenode:9000/data/title.basics.tsv
```

Figura 2.11: 1ª Tarefa : 2ª Alternativa - Transferência do ficheiro *"title.basics.tsv"* para a plataforma *Hadoop HDFS*

2.2.3 População da tabela *HBase*

Quanto à população da tabela criada previamente foi igualmente implementada uma classe *Java* para o efeito, designada por *Movie2Details*. Esta classe incorpora uma tarefa assente no paradigma *MapReduce*, onde é apenas elaborada a fase de *map*. Nessa mesma etapa é processada cada linha do ficheiro de *input* presente na plataforma *Hadoop HDFS* e, quando o tratamento estiver concluído, o resultado obtido é colocado na tabela *movies*.

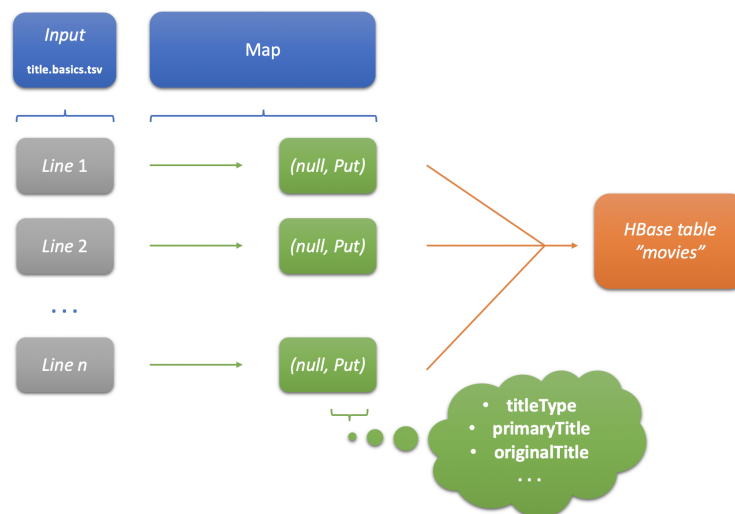


Figura 2.12: 1ª Tarefa - Esquema do paradigma *MapReduce*

2.3 2ª Tarefa

Tal como foi descrito no início do 2º capítulo deste documento, esta tarefa é composta por 3 alíneas distintas. Como tal é preciso tomar abordagens diferentes de forma a obter resultados corretos para cada uma das mesmas. Após uma leitura cuidadosa sobre o que é pedido em cada uma destas subtarefas, os elementos deste grupo notaram que iria ser necessário recolher 4 dos ficheiros que fazem parte do *dataset IMDB* para extrair os resultados pretendidos. Apresenta-se de seguida os 4 ficheiros escolhidos:

- *"name.basics.tsv"*: conjunto de dados característicos de um determinado ator, como por exemplo o seu nome, ano de nascimento, entre outros;
- *"title.basics.tsv"*: informação detalhada dos filmes, nomeadamente o ano de começo, géneros, entre outros;
- *"title.principals.tsv"*: conjunto de dados relativos aos atores que integram um determinado filme;
- *"title.ratings.tsv"*: informação associada à classificação e votação dos filmes presentes na plataforma *IMDB*.

Tal como se pode observar na listagem acima, o formato utilizado para estes ficheiros é o formato *.tsv*. A escolha desta extensão em detrimento de outras é exatamente a mesma que se encontra no início do subcapítulo relativo à 1ª tarefa.

À semelhança do que foi indicado nesse mesmo subcapítulo, a execução das soluções desenvolvidas nas secções 2.3.1 e 2.3.3 são realizadas com o auxílio do mesmo ficheiro *Dockerfile* com as mesmas opções de execução.

2.3.1 Criação da tabela *HBase*

De maneira a criar a tabela *HBase* associada a esta tarefa, foi implementada uma classe *Java*, ***CreateTableActors***, que, após conectar-se com a base de dados não relacional *HBase*, trata da sua criação e configuração. Durante esse processo, são produzidas duas famílias de colunas, denominadas por ***details*** e ***movies***, onde serão guardados todos os dados pertinentes para esta tarefa. Para além disso, tal como o nome da classe atrás evidencia, foi atribuído o nome ***actors*** à tabela criada.

Foi também criada uma classe *Java* extra, ***DeleteTableActors***, que trata de eliminar a tabela descrita anteriormente, sempre que for oportuno.

Apresenta-se de seguida o modelo da tabela *HBase* pretendido para a concretização desta tarefa:

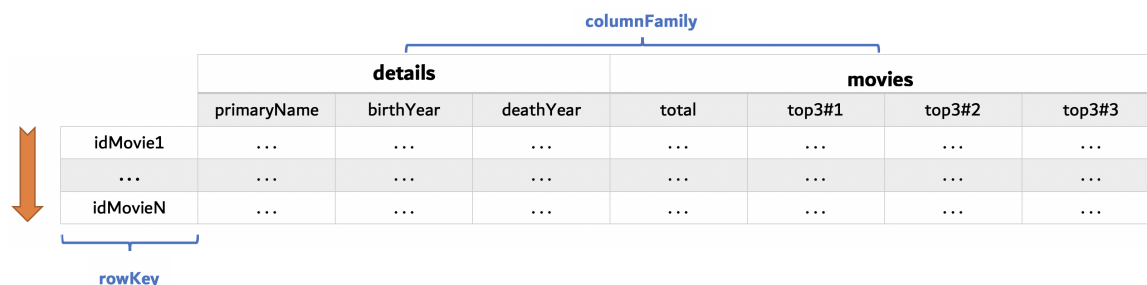


Figura 2.13: 2ª Tarefa - Modelo da tabela *HBase* "*actors*"

2.3.1.1 Alternativa

Uma possibilidade válida para realizar todo o processo associado à criação da tabela requerida seria utilizar a *HBase shell* de forma direta. Por forma a aceder a este recurso basta invocar a instrução presente na figura nº 6. Exibe-se de seguida o comando alternativo para realizar a criação da tabela mencionada:

```
hbase(main):001:0> create "actors", "details", "movies"
```

Figura 2.14: 2ª Tarefa : Alternativa - Criação da tabela *HBase "actors"*

Quanto à remoção da mesma tabela, à semelhança do procedimento tomado para a sua criação, adota-se a estratégia de usufruir explicitamente o mecanismo disponibilizado pela *HBase shell*:

```
hbase(main):001:0> disable "actors"
hbase(main):002:0> drop "actors"
```

Figura 2.15: 2ª Tarefa : Alternativa - Remoção da tabela *HBase "actors"*

2.3.2 Transferência de ficheiros para a plataforma *Hadoop HDFS*

Dado que o ficheiro *"title.basics.tsv"* já foi transferido para a plataforma *Hadoop HDFS* na tarefa anterior, resta apenas transferir os 3 ficheiros sobranceiros presentes no início deste subcapítulo (isto é, *"name.basics.tsv"*, *"title.principals.tsv"* e *"title.ratings.tsv"*). Antes de mostrar as duas alternativas para proceder à transferência destes 3 ficheiros, é importante salientar que a pasta criada na 1ª tarefa (denominada por *data*) é novamente utilizada para armazenar os mesmos. A instrução associada à criação da mesma pasta encontra-se disponível na figura nº 9.

2.3.2.1 1ª Alternativa

Nesta possibilidade evidencia-se o campo `source` que corresponde à diretoria da pasta que contém o ficheiro de *input* pretendido. Dito isto, apresenta-se agora a primeira alternativa:

```
docker run --network docker-hbase_default
--env-file docker-hbase/hadoop.env
--mount type=bind,source="/path/to/local/folder/data",target=/data
bde2020/hadoop-base hdfs dfs -put /data/name.basics.tsv /data

docker run --network docker-hbase_default
--env-file docker-hbase/hadoop.env
--mount type=bind,source="/path/to/local/folder/data",target=/data
bde2020/hadoop-base hdfs dfs -put /data/title.principals.tsv /data

docker run --network docker-hbase_default
--env-file docker-hbase/hadoop.env
--mount type=bind,source="/path/to/local/folder/data",target=/data
bde2020/hadoop-base hdfs dfs -put /data/title.ratings.tsv /data
```

Figura 2.16: 2ª Tarefa : 1ª Alternativa - Transferência dos 3 ficheiros de *input* para a plataforma *Hadoop HDFS*

2.3.2.2 2ª Alternativa

Esta opção corresponde ao modo interativo de execução disponibilizado pela instrução *docker run*. Uma vez feita esta observação, expõe-se a seguir a segunda alternativa:

```
docker run -it
    --network docker-hbase_default
    --env-file docker-hbase/hadoop.env
    bde2020/hadoop-base bash

curl https://datasets.imdbws.com/name.basics.tsv.gz | gunzip |
hdfs dfs -put - hdfs://namenode:9000/data/name.basics.tsv

curl https://datasets.imdbws.com/title.principals.tsv.gz | gunzip |
hdfs dfs -put - hdfs://namenode:9000/data/title.principals.tsv

curl https://datasets.imdbws.com/title.ratings.tsv.gz | gunzip |
hdfs dfs -put - hdfs://namenode:9000/data/title.ratings.tsv
```

Figura 2.17: 2ª Tarefa : 2ª Alternativa - Transferência dos 3 ficheiros de *input* para a plataforma *Hadoop HDFS*

2.3.3 População da tabela *HBase*

Ao contrário do que foi feito na primeira parte do trabalho, foi tomada a decisão de fazer uma separação do trabalho em duas unidades distintas: em primeiro lugar, é inserida a informação relativa aos detalhes pessoais de cada ator (nome e data de nascimento/morte); de seguida, faz-se a inserção dos dados relativos à carreira destes (número de filmes e melhores filmes).

2.3.3.1 Nome, datas de nascimento e morte do ator

Para a população da tabela criada com a informação relativa ao nome e data de nascimento/morte de cada ator, foi implementada uma classe *Java*, à qual foi dada a designação de ***Actor2Details***. Esta classe utiliza uma versão simplificada do paradigma *MapReduce*, pois só é executada a fase de *map*. Nesta fase única, são processadas as linhas do ficheiro *name.basics.tsv*, previamente inserido no *Hadoop HDFS*, escolhendo a informação relevante e inserindo o resultado pretendido na tabela *actors*.

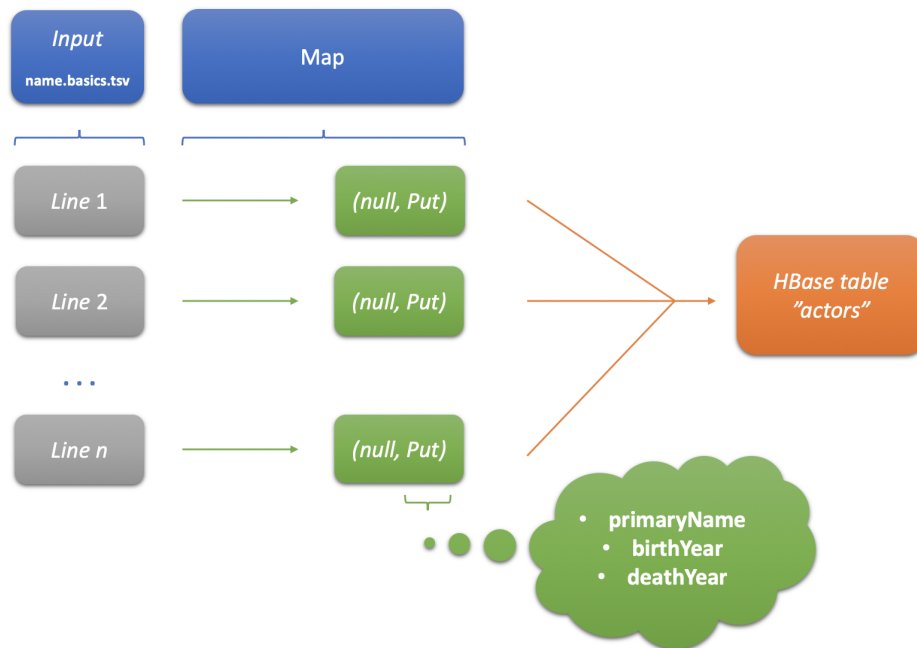


Figura 2.18: 2ª Tarefa - *Actor2Details* - Esquema do paradigma *MapReduce*

2.3.3.2 Número total de filmes + Top 3

Por último, para completar a informação relativa a cada ator, foi criada uma classe denominada ***Actor2Movies***. Esta classe consiste na execução de duas tarefas *MapReduce* sobre os dados dos ficheiros presentes no *Hadoop HDFS* e dos dados contidos na tabela gerada no primeiro exercício. Passando à primeira tarefa, esta tem como responsabilidade a geração de pares $(nconst, (originalTitle, averageRating))$, em que *nconst* é o identificador do ator, *originalTitle* representa o título de um filme em que esse ator participou e *averageRating* a classificação média desse mesmo filme. Para atingir este objetivo, são efetuadas três sub-tarefas *map*, a que são dados os nomes *Left*, *Middle* e *Right*, como é habitual em execuções deste tipo. Falando de cada uma em particular, temos que a *Left* toma como entrada o conteúdo do ficheiro *title.principals.tsv* e gera um par $(tconst, (L, nconst))$ para cada ator *nconst* que participou no filme *tconst*. A *Middle* extrai os dados da tabela criada no primeiro exercício, com a informação de cada filme, e cria pares $(tconst, (M, originalTitle))$, ou seja, a cada identificador de um filme associa o seu título original. Temos ainda a *Right*, que, a partir dos conteúdos do ficheiro *title.rating.tsv*, associa a cada identificador de um filme a respetiva classificação $((tconst, (R, averageRating)))$. Note-se que as letras L, M e R são prefixadas ao conteúdo das variáveis correspondentes. Como nesta fase de *map* a chave de cada par resultante é o identificador do filme *tconst*, quando se passa à fase *reduce* temos a certeza que, para um dado filme, o seu título, todos os atores que nele participaram e a sua classificação estarão associados à sua chave. Assim, o nosso *reducer* consiste na escrita, para um *sequence file*, de associações do tipo $(nconst, (originalTitle, averageRating))$, ou seja, para cada ator é introduzida uma destas linhas por cada filme em que participou.

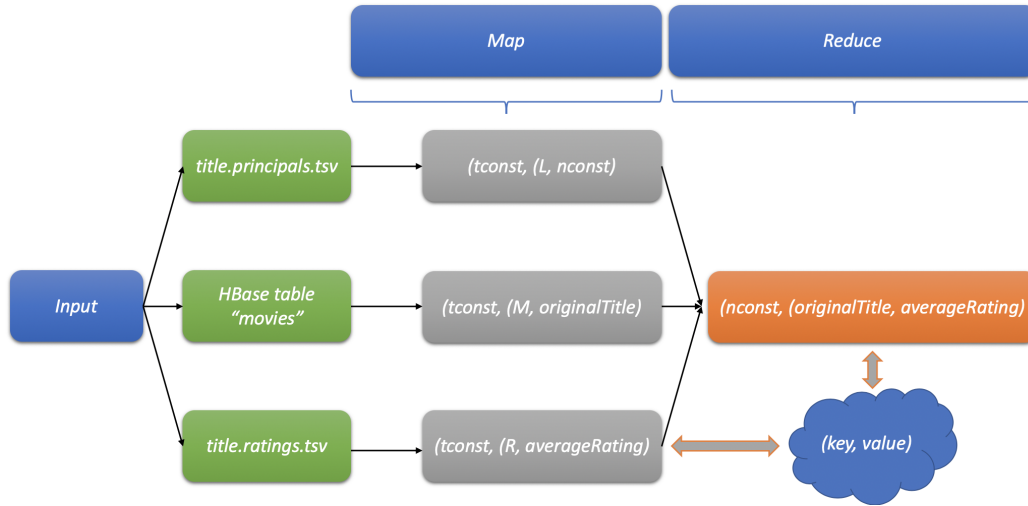


Figura 2.19: 2ª Tarefa - *Actor2Movies* - 1º Esquema do paradigma *MapReduce*

Estando esta informação armazenada no *HDFS* num *sequence file*, resta apenas executar uma outra tarefa *MapReduce* para inserir os dados na tabela "*actors*". Como os dados foram tratados na tarefa anterior, na fase de *map* é apenas necessário passar a informação tal como está para a fase de *reduce*. É aqui, tendo em conta que a cada ator está associada uma lista de pares $(originalTitle, averageRating)$, que será calculado o número de filmes de cada ator e a lista dos 3 melhores filmes. estando os dados previamente processados, isto torna-se trivial, sendo apenas necessário ver o comprimento desta lista para obter o número de filmes do ator e ordenar os mesmos por classificação, escolhendo os primeiros 3. Com vista a tornar a execução determinista, tivemos o cuidado de, em caso de empate na classificação, escolhermos os filmes por ordem alfabética. Para finalizar, os dados são inseridos na tabela pretendida.

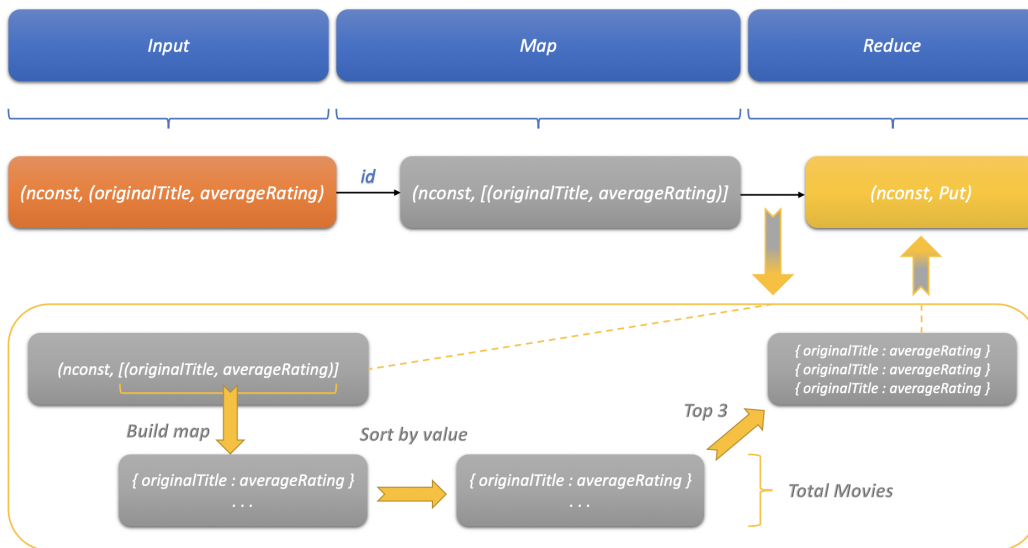


Figura 2.20: 2ª Tarefa - *Actor2Movies* - 2º Esquema do paradigma *MapReduce*

Capítulo 3

Conclusão

Para concluir, temos a referir que este trabalho foi bastante enriquecedor e muito provavelmente representativo de uma tarefa comum de Cientistas de Dados para processamento e armazenamento de informação resultante da observação de diversas fontes. Tivemos a oportunidade de utilizar o paradigma *MapReduce* do *Hadoop* e ainda o sistema de ficheiros *HDFS*. Estas ferramentas, por estarem orientadas para este tipo de tarefas, foram indicadas para este trabalho. Apesar disso, pudemos constatar alguns problemas, pois não nos é dada liberdade total nas operações que queremos efetuar sobre os dados. Um exemplo, é a imposição de termos de ter uma tarefa de *map*, mesmo que tal não seja necessário. Outro, é a obrigatoriedade de escrever os dados para disco entre duas tarefas. Como vimos nas aulas, desde a criação do *MapReduce* do *Hadoop* surgiu o *Spark*, que nos dá muito mais liberdade no tipo e ordem das operações que queremos realizar.

Apêndice A

Observações

- Documentação *Java* 8:
`https://docs.oracle.com/javase/8/docs/api/`
- *Maven*:
`https://maven.apache.org/`
- *Hadoop*:
`https://hadoop.apache.org/`
- *HBase*:
`https://hbase.apache.org/`