

# Gestão de Grandes Conjuntos de Dados

## 2º Trabalho Prático

-

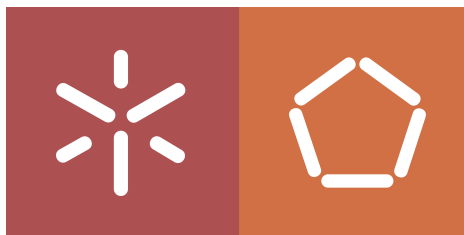
Mestrado em Engenharia Informática  
Universidade do Minho

### Grupo nº 8

---

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

30 de maio de 2020



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Implementação</b>	<b>4</b>
2.1	Configuração . . . . .	4
2.1.1	Arranque do <i>cluster</i> . . . . .	5
2.1.2	Execução de tarefas . . . . .	7
2.2	1ª Tarefa . . . . .	8
2.2.1	<i>Log</i> . . . . .	8
2.2.2	<i>Top3</i> . . . . .	9
2.2.2.1	Alternativa . . . . .	9
2.2.3	<i>Trending</i> . . . . .	9
2.2.3.1	Alternativa . . . . .	9
2.3	2ª Tarefa . . . . .	9
2.3.1	<i>Top10</i> . . . . .	9
2.3.1.1	Alternativa . . . . .	10
2.3.2	<i>Friends</i> . . . . .	10
2.3.2.1	Alternativa . . . . .	11
2.3.3	<i>Ratings</i> . . . . .	11
2.3.3.1	Alternativa . . . . .	11
2.4	3ª Tarefa . . . . .	11
<b>3</b>	<b>Conclusão</b>	<b>12</b>
<b>A</b>	<b>Observações</b>	<b>13</b>

# Lista de Figuras

2.1	Configuração - Criação da instância associada à entidade <i>master</i> . . . . .	5
2.2	Configuração - Criação da instância associada à entidade <i>worker1</i> . . . . .	5
2.3	Configuração - Criação da instância associada à entidade <i>worker2</i> . . . . .	5
2.4	Configuração - <i>swarm master</i> . . . . .	6
2.5	Configuração - <i>swarm worker1</i> . . . . .	6
2.6	Configuração - <i>swarm worker2</i> . . . . .	6
2.7	Configuração - Ativação do ambiente da entidade <i>master</i> . . . . .	6
2.8	Configuração - Verificação da existência das 3 entidades do sistema . . . . .	6
2.9	Configuração - Compilação do utensílio <i>streamgen</i> . . . . .	6
2.10	Configuração - Arranque do sistema . . . . .	6
2.11	Configuração - Ficheiro " <i>docker-compose.yml</i> " - <i>streamgen</i> . . . . .	7
2.12	Configuração - Verificação da disponibilidade do serviço relativo à rede <i>mystack_default</i> . . . . .	7
2.13	Configuração - Acesso à plataforma <i>Hadoop HDFS</i> . . . . .	7
2.14	Configuração - Obtenção do ficheiro <i>jar</i> do projeto . . . . .	7
2.15	Configuração - <i>Dockerfile</i> . . . . .	8
2.16	Configuração - Opções de execução do ficheiro <i>Dockerfile</i> . . . . .	8
2.17	2 <sup>a</sup> Tarefa ( <i>batch</i> ) - Esquema do processamento relativo à subtarefa <i>Top10</i> . . . . .	10
2.18	2 <sup>a</sup> Tarefa ( <i>batch</i> ) - Esquema do processamento relativo à subtarefa <i>Friends</i> . . . . .	11

# Capítulo 1

## Introdução

Neste trabalho prático é requerida a concretização e avaliação experimental de tarefas de armazenamento e processamento de dados através do uso da ferramenta computacional *Spark* (*batch* e *streaming*). Por forma a realizar estas tarefas, são utilizados os dados públicos do *IMDb*, que se encontram disponíveis em:

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

Para além destes dados, é também utilizado um gerador de *streams*, baseado nos mesmos, que simula uma sequência de votos individuais de utilizadores. Este utensílio foi desenvolvido pelo docente desta unidade curricular e encontra-se disponível na plataforma *Blackboard*.

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

## Capítulo 2

# Implementação

Para a realização com sucesso deste trabalho prático, é solicitada a elaboração de três tarefas. Apresentam-se de seguida as mesmas:

1. Desenvolver uma componente de processamento de *streams* que produza os seguintes resultados:
  - **Log**: armazenar todos os votos individuais recebidos, etiquetados com a hora de chegada aproximada ao minuto, em lotes de 10 minutos. Cada lote deve ser guardado num ficheiro cujo nome identifica o período de tempo;
  - **Top3**: exibir a cada minuto o *top* 3 dos títulos que obtiveram melhor classificação média nos últimos 10 minutos;
  - **Trending**: apresentar a cada 15 minutos os títulos em que o número de votos recolhido nesse período sejam superiores aos votos obtidos no período anterior, independentemente do valor dos votos.
2. Implementar uma componente de processamento em *batch* que permita realizar as seguintes tarefas:
  - **Top10**: calcular o *top* 10 dos atores que participaram em mais títulos diferentes;
  - **Friends**: computar o conjunto de colaboradores de cada ator (i.e., outros atores que participaram nos mesmos títulos);
  - **Ratings**: atualizar o ficheiro "*title.ratings.tsv*" tendo em conta o seu conteúdo anterior e os novos votos recebidos até ao momento.
3. Escolher a configuração e a implementação que, para o mesmo *hardware*, permite receber e tratar o maior débito de eventos. Esta tomada de decisão deve ser devidamente justificada com recurso a resultados experimentais.

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 Configuração

Nesta secção do relatório são apresentadas as configurações relativas ao arranque do *cluster*, instanciando-se tanto as respetivas entidades (*master*, *worker1* e *worker2*) como as máquinas virtuais pertencentes à plataforma *Google Cloud*, e, ainda, a configuração genérica para a execução individual de cada exercício proposto.

### 2.1.1 Arranque do *cluster*

A configuração escolhida para a realização deste projeto coincide com a que foi sugerida pelo docente desta unidade curricular, isto é, o ***docker swarm***. Esta configuração permite não só tirar partido da ferramenta computacional *Google Cloud* como também possibilita o uso de plataformas como o *Hadoop HDFS* e o *Apache Spark*. Como seria de esperar, todos os ficheiros de *input* utilizados para atingir os objetivos traçados neste trabalho prático são armazenados no sistema *Hadoop HDFS*.

Exibe-se de seguida todos os passos de configuração associados ao *docker swarm*:

1. criação das instâncias relativas às 3 entidades intrínsecas à arquitetura do sistema, isto é, as entidades *master*, *worker1* e *worker2*:

```
1 docker-machine create \  
2     --driver google --google-project ferrous-aleph-271712 \  
3     --google-zone europe-west1-b \  
4     --google-machine-type n1-standard-2 \  
5     --google-disk-size=100 \  
6     --google-disk-type=pd-ssd \  
7     --google-machine-image \  
8     https://www.googleapis.com/compute/v1/projects/centos-cloud/global/images/centos-7-v20200309 \  
9     master
```

Figura 2.1: Configuração - Criação da instância associada à entidade *master*

```
1 docker-machine create \  
2     --driver google --google-project ferrous-aleph-271712 \  
3     --google-zone europe-west1-b \  
4     --google-machine-type n1-standard-2 \  
5     --google-disk-size=100 \  
6     --google-disk-type=pd-ssd \  
7     --google-machine-image \  
8     https://www.googleapis.com/compute/v1/projects/centos-cloud/global/images/centos-7-v20200309 \  
9     worker1
```

Figura 2.2: Configuração - Criação da instância associada à entidade *worker1*

```
1 docker-machine create \  
2     --driver google --google-project ferrous-aleph-271712 \  
3     --google-zone europe-west1-b \  
4     --google-machine-type n1-standard-2 \  
5     --google-disk-size=100 \  
6     --google-disk-type=pd-ssd \  
7     --google-machine-image \  
8     https://www.googleapis.com/compute/v1/projects/centos-cloud/global/images/centos-7-v20200309 \  
9     worker2
```

Figura 2.3: Configuração - Criação da instância associada à entidade *worker2*

De salientar que a designação **ferrous-aleph-271712** corresponde ao identificador do projeto presente na plataforma *Google Cloud* de um dos elementos que compõem este grupo. Assim, esta denominação deve ser substituída pelo nome do projeto do utilizador em causa.

2. configuração do *swarm* relativo às entidades *master*, *worker1* e *worker2*:

```
1 docker-machine ssh master sudo docker swarm init
```

Figura 2.4: Configuração - *swarm master*

```
1 docker-machine ssh worker1 sudo docker swarm join --token \
2     SWMTKN-1-5zfy2iio54tma997pnt96gq5095fimqn2hxr2a8j16ogq0n3c9-0kp6mi5iuj956gpl9sfccd5bo\
3     10.132.0.8:2377
```

Figura 2.5: Configuração - *swarm worker1*

```
1 docker-machine ssh worker2 sudo docker swarm join --token \
2     SWMTKN-1-5zfy2iio54tma997pnt96gq5095fimqn2hxr2a8j16ogq0n3c9-0kp6mi5iuj956gpl9sfccd5bo\
3     10.132.0.8:2377
```

Figura 2.6: Configuração - *swarm worker2*

### 3. ativação do ambiente da entidade *master*:

```
1 docker-machine env master
2 eval $(docker-machine env master)
```

Figura 2.7: Configuração - Ativação do ambiente da entidade *master*

### 4. verificação da existência das 3 entidades presentes no sistema e das respectivas propriedades:

```
1 docker node ls
```

Figura 2.8: Configuração - Verificação da existência das 3 entidades do sistema

### 5. compilação do gerador de *streams*, isto é, o *streamgen*:

```
1 mvn package
2 docker build -t streamgen .
```

Figura 2.9: Configuração - Compilação do utensílio *streamgen*

### 6. arranque do sistema com a configuração *swarm* especificada:

```
1 docker stack deploy -c ../swarm-spark/docker-compose.yml mystack
```

Figura 2.10: Configuração - Arranque do sistema

É de realçar que no ficheiro "*docker-compose.yml*" encontra-se uma configuração *docker* semelhante à que foi utilizada no guião nº 8 desta unidade curricular. A única diferença presente no mesmo diz respeito à integração do gerador de *streams* (*streamgen*) desenvolvido pelo docente como um *container docker* do sistema. Assim, no momento do arranque do sistema, a ferramenta *streamgen* é convenientemente invocada, ficando à espera de novas conexões por parte dos utilizadores. Apresenta-se de seguida a configuração presente no ficheiro "*docker-compose.yml*" relativa ao *streamgen*:

```

1 streamgen:
2   image: streamgen
3   command: hdfs:///data/title.ratings.tsv 120
4   env_file:
5     - ./hadoop.env
6   deploy:
7     mode: replicated
8     replicas: 1
9     placement:
10      constraints:
11        - "node.role==manager"

```

Figura 2.11: Configuração - Ficheiro "*docker-compose.yml*" - *streamgen*

## 7. verificação da disponibilidade do serviço relativo à rede *mystack\_default*:

```

1 docker stack ls
2 docker service ls
3 docker network ls

```

Figura 2.12: Configuração - Verificação da disponibilidade do serviço relativo à rede *mystack\_default*

## 8. acesso à plataforma *Hadoop HDFS* por parte do utilizador:

```

1 docker run --network mystack_default --env-file ../swarm-spark/hadoop.env -it bde2020/hadoop-base \
2   bash

```

Figura 2.13: Configuração - Acesso à plataforma *Hadoop HDFS*

Com a concretização dos 8 passos descritos acima fica concluída a configuração associada à ferramenta *docker swarm*.

### 2.1.2 Execução de tarefas

Para a execução dos 6 exercícios descritos na secção relativa à implementação, é disponibilizado um ficheiro com a designação *Dockerfile*. Neste é especificado o nome da classe relativa à proposta que se presente executar, procedendo corretamente à sua invocação. Como tal, em primeiro lugar é preciso desempenhar a seguinte instrução:

```

1 mvn package

```

Figura 2.14: Configuração - Obtenção do ficheiro *jar* do projeto



De seguida, invoca-se o *dockerfile* em causa. Expõe-se agora o seu conteúdo:

```
1 FROM bde2020/spark-base:2.4.4-hadoop2.7
2 COPY target/TP2-1.0-SNAPSHOT.jar /
3 ENTRYPOINT ["/spark/bin/spark-submit", "--class", "package.className", "--master", \
4             "spark://spark-master:7077", "/TP2-1.0-SNAPSHOT.jar"]
```

Figura 2.15: Configuração - *Dockerfile*

A este ficheiro estão associadas alguma opções de execução que garantem uma comunicação fidedigna com o sistema inicialmente declarado. Eis as mesmas:

```
1 -p 4040:4040 --network mystack_default --env-file ../swarm-spark/hadoop.env
```

Figura 2.16: Configuração - Opções de execução do ficheiro *Dockerfile*

De salientar que é escolhida a opção *master* no campo relativo ao *docker machine* para, mais uma vez, garantir uma configuração válida do sistema em causa.

## 2.2 1ª Tarefa

Na 1ª tarefa deste projeto é pedido o desenvolvimento de uma componente de processamento de *streams*. Nesta é solicitada a realização de vários exercícios com diferentes características. Como tal, apresenta-se de seguida as respetivas implementações e, ainda, possíveis alternativas nas suas concretizações.

### 2.2.1 Log

Neste exercício, tal como foi indicado anteriormente neste capítulo, é imposto o armazenamento de todos os votos individuais recebidos com a indicação da hora de chegada aproximada ao minuto, em lotes de 10 minutos, sendo que cada lote é guardado com o nome relativo ao período de tempo em causa.

Dito isto, para proceder ao tratamento da aproximação da hora recebida ao minuto, foi implementada uma função para o efeito. O grupo optou por não só guardar a hora referida como também a data em questão, exibindo, desta forma, um maior detalhe da informação recebida. Uma vez implementada esta função, seguindo a sugestão fornecida pelo docente desta unidade curricular, foi utilizado o método *transform* que faz uso da noção de tempo. Com esta vertente, é possível associar a cada *rdd* do conjunto de dados o tempo exato em que este foi processado.

Por fim, com os aspetos computacionais mencionados acima, resta armazenar os dados recebidos pela ferramenta *streamgen* com recurso à definição de uma janela que respeite o enunciado deste exercício, isto é, 10 minutos de duração e 10 minutos de deslocamento. Os *rdd's* em causa são guardados na diretoria `Log/Lot{i}` - `dd-MM-yyyy HH-mm`, sendo que *i* diz respeito ao número do lote (que começa no número 1 e que é constantemente incrementado) e `dd-MM-yyyy HH-mm` refere-se ao padrão do período temporal. De maneira a coletar num só ficheiro (*part-00000*) todos os *rdd's* referentes a este exercício, foi aplicado o método *coalesce(1)*. Esta última função, quando comparada com a *repartition()*, não necessita de efetuar um *shuffle* completo dos dados e, para além disso, é mais otimizada para a redução do número de partições. Para além disto, esta abordagem foi considerada nesta implementação

dado que no exercício *Ratings (batch)* será necessário realizar a leitura do ficheiro mencionado. Consequentemente, em vez de operar várias leituras de ficheiros associados a *rdd's*, efetua-se apenas uma leitura, permitindo, assim, uma maior eficiência.

### 2.2.2 *Top3*

#### 2.2.2.1 Alternativa

### 2.2.3 *Trending*

#### 2.2.3.1 Alternativa

## 2.3 2ª Tarefa

Na 2ª tarefa deste trabalho prático é proposta a implementação de uma componente de processamento em *batch*. Nesta é solicitada a realização de 3 exercícios com diferentes características e, como tal, foram adotadas abordagens distintas para cada um deles. Divulga-se de seguida as respetivas implementações.

### 2.3.1 *Top10*

Tal como foi mencionado no início 2º capítulo, nesta sub tarefa é pedido o cálculo dos 10 atores que participaram em mais filmes distintos.

Durante o processamento inicial do ficheiro "*title.principals.tsv*" é, tal como seria de esperar, ignorado o respetivo cabeçalho. Posteriormente, é extraída, linha após linha, a informação pertinente do mesmo, isto é, os identificadores do filme e do ator em questão, agrupando os dados pela segunda componente. Esta última ação é efetuada com recurso à chamada do método *groupByKey*. Uma vez realizada esta computação, obtém-se para cada ator a lista de filmes em que este participou. Atendendo ao resultado exigido neste exercício, basta, nesta etapa do processamento, efetuar a contagem dos filmes associados a cada ator, filtrando os 10 registos com maiores valores.

A recolha dos 10 atores que participaram em mais filmes é formalizada com a chamada do método *top*. Esta função permite extrair os *k* maiores registos de um *RDD* segundo uma determinada ordem. Para o caso deste exercício, houve a necessidade de implementar um comparador explícito, numa classe à parte, dado que o tipo de dados *Tuple2* não é, por definição, serializável.

Tendo em consideração este último detalhe, conclui-se a realização desta sub tarefa.

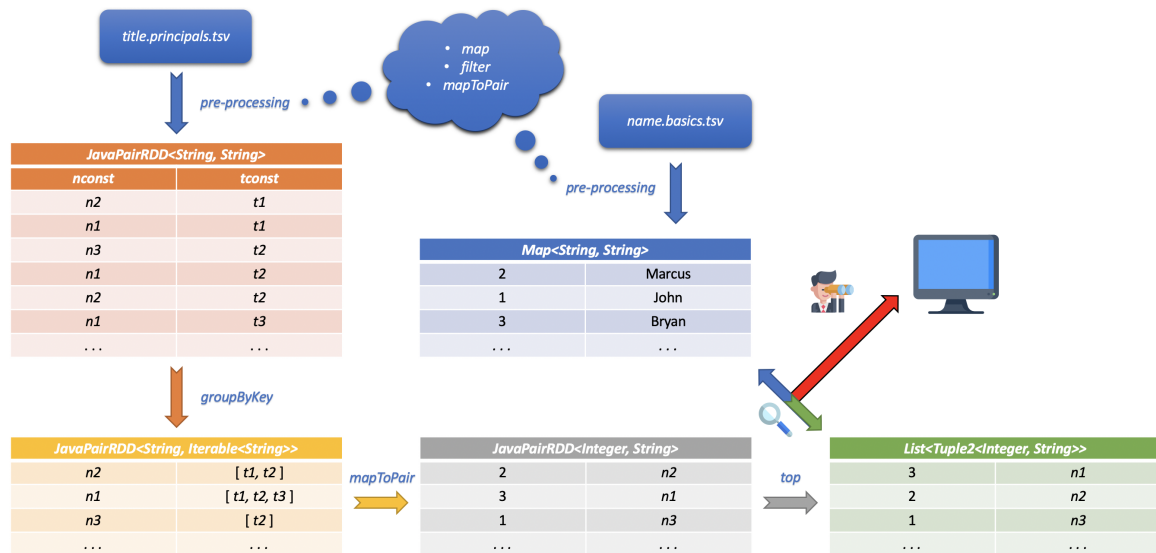


Figura 2.17: 2ª Tarefa (batch) - Esquema do processamento relativo à sub tarefa *Top10*

### 2.3.1.1 Alternativa

Uma forma alternativa de resolver este exercício seria, na última fase do processamento, utilizar o método *take* em detrimento da função *top*. Esta escolha não foi tomada em consideração na implementação uma vez que o primeiro método necessita previamente que a informação esteja devidamente ordenada. Esta ordenação teria de ser realizada com a invocação do método *sortByKey(false)*, colocando a contagem dos filmes em que cada ator participou de forma decrescente. Este último facto representa uma ineficiência no cálculo do resultado pretendido uma vez que é efetuada a ordenação completa da informação em causa e, para além disso, realiza-se desnecessariamente um passo computacional extra.

### 2.3.2 Friends

Neste exercício é requerido a computação do conjunto de colaboradores associado a cada ator, ou seja, o grupo dos atores que participam nos mesmos filmes.

Durante o processamento inicial do ficheiro *"title.principals.tsv"* é, tal como seria de esperar, ignorado o respetivo cabeçalho. Posteriormente, é extraída, linha após linha, a informação pertinente do mesmo, isto é, os identificadores do filme e do ator em questão, agrupando os dados pela primeira componente. Esta última ação é efetuada com recurso à chamada do método *groupByKey*. De forma a obter o resultado solicitado nesta sub tarefa, é necessário, nesta fase da computação, proceder à realização de uma operação denominada por produto cartesiano. Nesta operação computa-se, num dado momento, vários pares de atores que coloboraram num determinado filme. Uma vez realizado este cálculo, é invocado novamente o método *groupByKey* de forma a obter o resultado pretendido, isto é, o conjunto de colaboradores para cada ator presente nos dados públicos do *IMDb*.

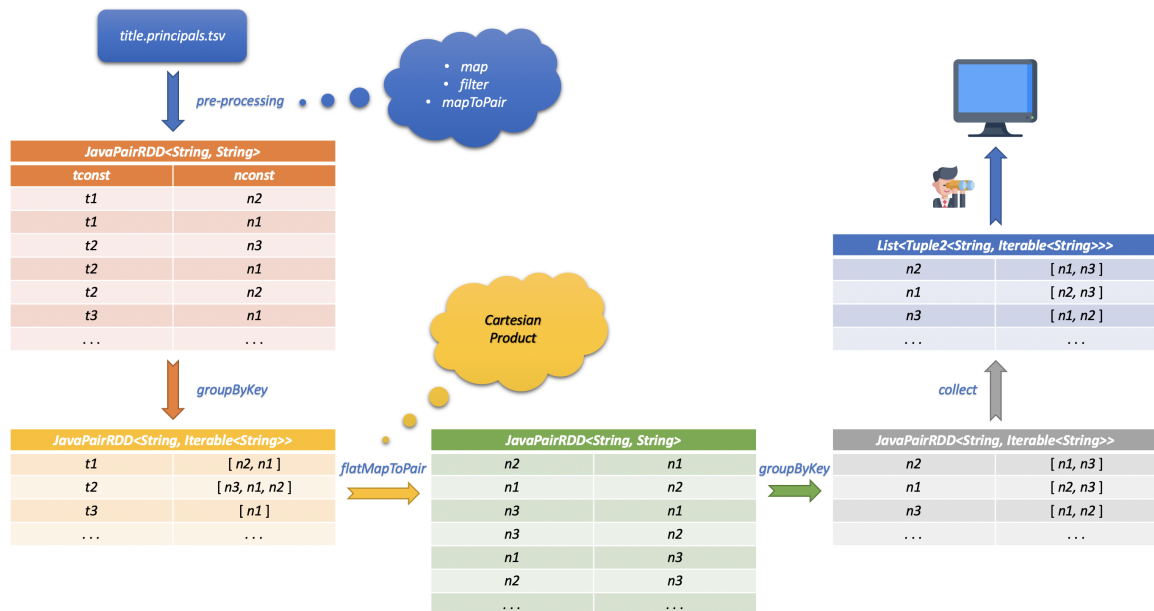


Figura 2.18: 2ª Tarefa (batch) - Esquema do processamento relativo à subtarefa *Friends*

### 2.3.2.1 Alternativa

## 2.3.3 Ratings

### 2.3.3.1 Alternativa

## 2.4 3ª Tarefa

## Capítulo 3

## Conclusão

# Apêndice A

## Observações

- Documentação *Java* 8:  
`https://docs.oracle.com/javase/8/docs/api/`
- *Maven*:  
`https://maven.apache.org/`
- *Docker*:  
`https://www.docker.com/`
- *Apache Spark*:  
`https://spark.apache.org/`
- *Apache Hadoop*:  
`http://hadoop.apache.org/`