Universidade do Minho

Mestrado Integrado em Engenharia Informática

Laboratórios de Informática III

Relatório do Projeto em Java

Grupo 54

André Amaral (a77456)
Ricardo Ponte (a79097)
Daniela Fernandes (a73768)

12 de Junho de 2018

Conteúdo

1 Introdução

"O Stack Overflow é, atualmente, uma das comunidades de perguntas e respostas mais utilizadas por desenvolvedores em todo o mundo. Nesta plataforma, qualquer utilizador pode colocar questões que serão depois respondidas por outros utilizadores. Segundo informação oficial, a plataforma tem cerca de 8.4 milhões de utilizadores e já se colocaram cerca de 15 milhões de questões, às quais se responderam 24 milhões de vezes. A informação resultante da utilização da plataforma é extremamente útil e valiosa, não só devido ao conteúdo das perguntas e respostas mas também aos metadados originados a partir dos mesmos."

No âmbito da unidade curricular de Laboratórios de Informática III, foi proposto um projeto na linguagem de programação Java em que o objetivo é fazer parse de um dump do Stack Overflow para armazenar e processar os dados com o intuito de realizar 11 interrogações propostas pela equipa docente. Relativamente a este aspeto foi usado para o parse a API SAX para processar ficheiros XML. Ao longo deste relatório serão explicadas as estratégias utilizadas para a resolução das interrogações.

2 Conceção da Solução

2.1 Classes Utilizadas

Tendo já realizado este projeto na linguagem C, definir as classes, as respetivas varíaveis de instância e métodos de classe, foi mais simples, pois já havia sido feito algo similar com as estruturas de dados desenvolvidas anteriormente.

2.2 Parse

Para a realização do parsing dos ficheiros XML, foi utilizada a API SAX (Simple API for XML). Esta API foi escolhida por ser eficiente a ler ficheiros XML e como é uma Streaming Interface for XML, a memória necessária para a realização do parsing é menor, conseguindo, desta forma, realizar um output mais rápido em relação a outras API do estilo DOM. Para usar esta API foi necessário desenvolver classes que seriam *Event Handler*, cujo o objetivo foi a recolha de informação dos ficheiros XML.

2.2.1 PostHandler

```
public class PostHandler extends DefaultHandler {
    private Map<Long,Post> hashPost;
}
```

O *PostHandler* serve para carregar a informação de todos os posts contidos no ficheiro XML(Posts.xml). Em cada evento XML que ocorra no ficheiro, será invocado o método *startElement* e para cada elemento em XML que seja inicializado por *row*, irá obter os atributos pretendidos. De seguida, será verificado se o post se trata de uma questão ou de uma resposta, através do seu *typeID*. Caso seja uma questão, serão armazenados todos os dados referentes à mesma num objecto *Question* que depois será adicionado ao *hashPost*. Se o post for uma resposta, o processo anterior será repetido e adaptado para uma resposta.

2.2.2 TagHandler

```
public class TagHandler extends DefaultHandler {
    private Map<Long, Tag> hashTag;
}
```

O *TagHandler* serve para carregar a informação de todos as tags contidas no ficheiro XML(Tags.xml). Em cada evento XML que ocorra no ficheiro, será invocado o método *startElement* e para cada elemento em XML que seja inicializado por *row*, irá obter os atributos pretendidos. Por fim, guarda esses atributos no objeto *Tag* que depois será inserido no *hashTag*.

2.2.3 UserHandler

```
public class UserHandler extends DefaultHandler {
    private Map<Long,User> hashUser;
}
```

O *UserHandler* serve para carregar a informação de todos os utilizadores contidas no ficheiro XML(Users.xml). Em cada evento XML que ocorra no ficheiro, será invocado o método *startElement* que sempre que um elemento em XML seja inicializado por *row*, irá obter os atributos pretendidos. Por fim, guarda esses atributos no objeto *User* que depois será inserido no *hashUser*.

2.3 User

```
public class User {
    private long id;
    private int reputation;
    private String displayname;
    private String shortbio;
    private int postcount;
}
```

A class *User* representa os utilizadores do StackOverflow. Na figura acima estão representadas as variáveis de instância que correspondem a um utilizador. O *id* é o identificador, a *reputation* é a reputação, o *displayname* corresponde ao nome do utilizador, o *shortbio* é uma pequena bibliografia do utilizador. Por fim, foi adicionada a variável de instância *postcount* que serve como um contador para os post que cada utilizador faz.

2.4 Post

```
public class Post {
    private long id,ownerid;
    private int typeid,score,commentcount;
    private LocalDate date;
}
```

A classe *Post* representa os posts do StackOverflow.Na figura acima estão representadas as variáveis de instância que correspondem a um post.O *id* é o identificador do post, o *ownerid* é o identificador do criador do post, um *typeid* representa o tipo de post (pergunta ou resposta), um *score* é a pontuação do post, um *commentcount* é um contador de comentários e, por fim, uma *date* que representa a data em que o post foi efetuado. Esta classe será uma super classe pois todos os seus atributos são partilhados entre perguntas e respostas.

2.4.1 Question

```
public class Question extends Post {
        private int answercount;
        private String title;
        private String tags;
}
```

Esta subclasse *Question* representa uma questão. Para além dos atributos que herda do *Post* uma questão terá como variáveis de instância, o *answercount* correspondente ao número de respostas, um *title* que representa o título da questão e, por fim, a String *tags* em que estão contidas as tags utilizadas na questão.

2.4.2 Answer

```
public class Answer extends Post {
        private int parentid;
}
```

Esta subclasse *Answer* representa uma resposta. Para além dos atributos que herda do *Post* uma resposta conterá o *parentid* correspondente ao identificador da pergunta respondida.

2.5 Tag

```
public class Tag {
    private long id;
    private String name;
    private int counter;
}
```

A class *Tag* representa uma tag. Como demonstra a figura, foram utilizadas 3 variáveis de instância para representar as tags, sendo estas, o *id* que é o identificador, o *name* que é o nome da tag e, por fim, o *counter* que foi criado para a resolução da interrogação 11.

2.6 Interrogações

2.6.1 Interrogação 1

> A interrogação 1 retorna o título do post e o nome de utilizador do autor. Se o post for uma resposta, a função deverá retornar informações da pergunta correspondente.

O primeiro passo na resolução desta interrogação foi obter o post com o identificador fornecido. De seguida, foi obtido o identificador do criador do post de modo a rastrear o utilizador. Após o rastreamento, o seu nome irá ser inserido numa String. Relativamente ao post, é necessário saber se é pergunta ou resposta. Caso seja pergunta, o seu título será inserido numa String, caso contrário, a questão que é respondida será rastreada e o seu título será inserido noutra String. Por fim, é devolvido um novo objecto *Pair* que contém o par de Strings, sendo elas o título da pergunta e o nome do utilizador, respetivamente.

2.6.2 Interrogação 2

> A interrogação 2 retorna os identificadores dos N utilizadores com o maior número de posts de sempre (Considerando perguntas e respostas).

Como os utilizadores já se encontram ordenados pelo seu *PostCount*, nesta interrogação, apenas foi necessário adicionar os identificadores dos N primeiros utilizadores a uma lista e retornar a mesma.

2.6.3 Interrogação 3

> A interrogação 3 retorna o número total de posts entre um intervalo de tempo arbitrário em que as perguntas e as respostas estão identificadas separadamente.

Inicialmente foi necessário obter e filtar todos os posts que estão entre um intervalo de tempo e colocá-los todos numa lista. De seguinda, foram filtrados em duas listas diferentes as perguntas e as respostas e, por fim, é retornado um objecto *Pair* com o tamanho das duas listas anteriores.

2.6.4 Interrogação 4

> A interrogação 4 retorna uma lista com os identificadores de todas as perguntas que contêm uma determinada tag (entre um determinado intervalo de tempo), por ordem cronologicamente inversa.

Para a resolução desta interrogação, foi necessário obter e filtrar todos os posts que estão entre um intervalo de tempo e colocá-los todos numa lista. De seguida, foram filtrados numa lista todos os posts que são perguntas e que contêm uma determinada tag. Finalmente, é retornado numa lista os identificadores das perguntas anteriormente obtidas.

2.6.5 Interrogação 5

> A interrogação 5 retorna a informação do perfil de um utilizador (short bio) e os identificadores dos seus 10 últimos posts (ordenados por cronologia inversa).

Inicialmente, todos os posts do utilizador são carregados numa lista e a *shortbio* do utilizador é inserida numa String. A lista é percorrida apenas dez vezes, uma vez que, os posts já se encontram ordenados por cronologia inversa no *LinkedHashMap* e, em cada uma destas travessias, o identificador do post é adicionado a uma lista auxiliar. No fim, é retornado um *Pair* com a *shortbio* e a lista anteriormente obtida.

2.6.6 Interrogação 6

> A interrogação 6 retorna os identificadores das respostas com mais votos, em ordem decrescente do score (a diferença entre UpMod e DownMod) e estejam contidos num intervalo de tempo arbitrário.

A resolução desta interrogação torna necessário obter e filtrar todas as respostas que estão entre um intervalo de tempo. Após esse procedimento é usado um *Comparator* para ordenar as respostas de forma inversa do score. A lista com as respostas ordenadas é percorrida N vezes e em cada iteração será adicionado o identificador da resposta a uma lista auxiliar. Por fim, é retornada essa lista.

2.6.7 Interrogação 7

> A interrogação 7 retorna os identificadores das perguntas com mais respostas, por ordem decrescente do número de respostas.

Esta interrogação torna necessário obter e filtrar todas as perguntas que estão entre um intervalo de tempo.Após esse procedimento, é usado um *Comparator* para ordenar as perguntas de forma inversa da contagem de respostas. A lista com as questões ordenadas é percorrida N vezes e em cada iteração será adicionado o identificador da pergunta a uma lista auxiliar.Por fim, é retornada essa lista.

2.6.8 Interrogação 8

> A interrogação 8 retorna uma lista com os identificadores das perguntas cujos os títulos a contenham, ordenadas por cronologia inversa.

Inicialmente, foi necessário obter e filtrar todas as questões. De seguida, a lista das questões é filtrada dependendo se contem a palavra no seu título. A lista anteriormente obtida é percorrida N vezes e em cada iteração foi adicionado o identificador da pergunta a uma lista auxiliar que no fim é retornada.

2.6.9 Interrogação 9

A interrogação 9 retorna os identificadores das últimas perguntas em que participaram dois utilizadores específicos, por cronologia inversa.

Com o intuito de executar esta interrogação eficientemente, foram obtidos e filtrados todos os posts que ambos os utilizadores fizeram. De seguida, foi necessário comparar todos os posts obtidos anteriormente para saber se os utilizadores contribuiram para o mesmo post. Uma contribuição, tal como o projeto em C, foi definida como sendo:

1. Pergunta feita pelo utilizador1 e respondida pelo utilizador2;
2. Vice-versa do primeiro caso;
3. Ambos responderam à mesma pergunta.

Cada post foi analizado da seguinte forma. Caso o post seja uma pergunta feita por um dos utilizadores, foi verificado se o outro utilizador respondeu, adiciona-se o post a uma lista auxiliar caso isso se verifique e incrementa-se o contador. Se estivermos perante uma resposta é verificado se ambos os utilizadores responderam e caso isso aconteça esse post será adicionado a uma lista auxiliar e o seu contador incrementado.Quando o contador fica igual a N o ciclo acaba e a função retorna uma lista de identificadores dos posts pertencentes à lista auxiliar anterior.

2.6.10 Interrogação 10

A interrogação 10 devolve o identificador da melhor resposta (dado uma pergunta).

Tendo em vista a execução correta e eficiente desta interrogação, foram obtidos e filtrados todos os posts que são respostas de uma certa pergunta. Em seguinda, é percorrida a lista das respostas anteriormente obtidas e a cada resposta, é calculada a sua pontuação. A pontuação máxima será armazenada numa variável que sempre que for mudada, atualiza o seu valor e o identificador pretendido,que também está armazenado numa variável. Por fim, é retornada essa variável.

2.6.11 Interrogação 11

A interrogação 11 retorna os identificadores das tags mais usadas pelos utilizadores com melhor reputação (por ordem decrescente do número de vezes que a tag foi usada), num dado intervalo de tempo arbitrário.

No inicio da execução, é criado uma lista com os N primeiros utilizadores do *LinkedHashMap*. Em seguida, a lista anteriormente obtida é percorrida de modo a gerar todas as questões colocadas pelos N melhores utilizadores entre um dado intervalo de tempo. Esta lista será percorrida e a cada iteração, as suas tags serão divididas numa lista de Strings, serão rastreadas no *HashMap* das tags e posteriormente, o seu contador será incrementado e serão adicionadas a um novo *HashMap* auxiliar.O *HashMap* auxiliar é posteriormente convertido para uma lista de tags. Por fim, é usado um *Comparator* para ordenar a lista de tags de forma inversa do seu *counter*, ou caso este seja igual, por ordem crescente do seu identificador.No final da execução,são retornados os identificadores da lista obtida anteriormente.

3 Otimização de desempenho

No que toca a estratégias de desempenho foram utilizadas algumas no decorrer deste projeto. Para a resolução da interrogação 2 foi criada uma variável de instância na classe *User* (postcount) que será atualizada após o parse dos posts. Quando é feito o parse dos utilizadores, é percorrido o *hashPost* e o *postcount* do criador do post será incrementado. Isto será feito para todos os posts, atualizando assim o *Map* dos utilizadores. De seguida, será obtida uma lista de utilizadores ordenada por ordem decrescente do postcount. Por fim, a lista anteriormente obtida será inserida no *LinkedHashMap* dos utilizadores. Foi utilizado um *LinkedHashMap* para que a ordem de inserção permanecesse inalterada. Isto permite que a interrogação 2 tenha um tempo de execução muito reduzido, uma vez que apenas será necessário percorrer o *hashUser* N vezes. Coincidentemente, os utilizadores com melhor reputação têm também os maiores valores de *postcount*. Sendo assim, a ordenação por *postcount* não interfere com o output da interrogação 11, permitindo uma abordagem inicial semelhante à interrogação 2.

Para a resolução das outras interrogações, foram utilizadas *Streams*, verificando-se assim um aumento significativo da performance do programa.

```
Tempos com otimização:          |          Tempos sem otimização:
LOAD -> 13429 ms                |          LOAD -> 12730 ms
Query 1 -> 1 ms                 |          Query 1 -> 0 ms
Query 2 -> 0 ms                 |          Query 2 -> 266 ms
Query 3 -> 44 ms                |          Query 3 -> 133 ms
Query 4 -> 35 ms                |          Query 4 -> 193 ms
Query 5 -> 8 ms                 |          Query 5 -> 18 ms
Query 6 -> 28 ms                |          Query 6 -> 54 ms
Query 7 -> 21 ms                |          Query 7 -> 47 ms
Query 8 -> 36 ms                |          Query 8 -> 49 ms
Query 9 -> 54 ms                |          Query 9 -> 23 ms
Query 10 -> 12 ms               |          Query 10 -> 21 ms
Query 11 -> 55 ms               |          Query 11 -> 242 ms
CLEAN -> 6 ms                   |          CLEAN ->  5 ms
```

Como podemos ver pela figura, embora a load seja mais lenta após a otimização, os tempos de execução das interrogações melhoraram significativamente devido às otimizações efetuadas.

4 Conclusão

Após a realização do trabalho prático na linguagem de programação C, abordar o mesmo projeto numa linguagem de mais alto nível, Java, revelou ser mais simples, tanto pelo facto de já existir o plano do projeto anteriormente estruturado, como pelo facto de Java ser uma linguagem que permite mais liberdade na escolha de estruturas de dados, pelo facto de já estarem previamente implementadas.

Durante a conceção das diversas interrogações, uma das maiores dificuldades foi conseguir perceber e utilizar eficientemente as streams que o Java 8 disponibiliza, mas apesar do ceticismo inicial o grupo conseguiu rápidamente entender e utilizar, tanto eficientemente como indutivamente.

Comparativamente ao projeto em C, o grupo estima que houve uma redução do tempo de exeução em cerca de 8 segundos, para além do facto de não haver a preocupação com os *memory leaks* uma vez que o Java tem um sistema de *garbage collection* que não permite a ocorrência de *memory leaks*.

A programação deste projeto foi mais intuitiva no paradigma orientado a objetos, uma vez que toda a informação pode ser representada e contida em objetos.