Laboratórios de Informática III - Projecto em C

Mestrado Integrado em Engenharia Informática
Universidade do Minho

Bruno Martins (A80410), Leonardo Neri (A80056), Márcio Sousa (A82400)

I. Introdução

Este projeto teve como principal foco o processamento e tratamento de grandes volumes de dados, provenientes de um *dump* do *Stack Exchange*, com o objetivo de responder da forma mais eficiente possível a 11 queries previamente definidas pelos docentes da cadeira.

II. Análise do Problema

A base do problema foi reduzida a duas partes: a primeira parte consistiu em encontrar a melhor forma de armazenar em memória todos os dados resultantes do processamento dos ficheiros necessários do *dump* de forma a minimizar o máximo possível o tempo de *load* e o tempo de acesso aos dados, recorrendo para isso às estruturas de dados que parecessem mais apropriadas para esse fim. Para a segunda parte, o objetivo pretendido consistiu em responder às queries, de forma correta, no mínimo tempo possível, fazendo uso dos melhores algoritmos pensados pelos integrantes do grupo.

III. Método de Parsing

Começando pelo método de *parsing* dos ficheiros XML que foram objeto de análise, após alguma pesquisa foi decidido que seria usado o **SAX** (**S**imple **A**PI for **X**ML) em alternativa ao método **DOM** (**D**ocument **O**bject **M**odel), devido ao facto de o método **SAX** ser melhor quando se lida com ficheiros de grande volume, como é o caso. O método **SAX** faz com que tenhamos, em cada instante, uma ínfima parte do ficheiro em memória, enquanto que o **DOM** carrega tudo para memória, o que é desnecessário.

IV. Estruturas de Dados

Relativamente às estruturas de dados, foi dada especial importância ao encapsulamento, para garantir o acesso controlado aos dados guardados nas mesmas, para que estes nao pudessem ser alterados do exterior. Cada estrutura está definida num ficheiro .c juntamente com funções que nos permitem obter os dados necessários à resolução das queries, mantendo o encapsulamento dos dados. Cada estrutura tem também um ficheiro .h correspondente, onde estão as assinaturas das funções de acesso à estrutura. O grupo decidiu recorrer às estruturas pré-definidas da biblioteca GLIB, devido ao facto de estas apresentarem vantagens que serão mencionadas a seguir.

A. Posts, Users e Tags

Foi acordado pelos membros do grupo que, para armazenamento dos dados obtidos dos ficheiros XML relativos aos Posts, aos Users e às Tags, seriam criadas estruturas para cada um desses tipos de dados com os campos necessários para dar resposta às queries, e cada instância dessas estruturas seria posteriormente guardada numa Hash Table da GLIB, devido ao reduzido número de colisões que estas apresentam, e à facilidade de acesso aos dados que as Hash Tables proporcionam, devido ao facto de a *key*, juntamente com a função de hash, determinar exatamente a posição da Hash Table em que o elemento ficará, sendo extremamente mais eficiente procurar um elemento na Hash Table do que se estes fossem guardados em Arrays por exemplo, o que implicaria, na maioria das vezes, percorrer milhares de posições do Array, antes de encontrar o se procurava.

B. Post's Date

Devido ao facto de tantas queries pedirem algo entre duas datas, foi determinado que seria muito pouco eficiente "percorrer"toda uma Hash Table para ver a data de cada Post, e foi também notado, após análise do ficheiro XML com recurso a funções auxiliares, que por motivo desconhecido, a ordenação dos Posts sob a lógica "quanto menor o Post ID, mais antigo será o Post"não era respeitada em alguns casos, o que, juntamente com o facto de ser muito pouco eficiente, eliminou de imediato a hipótese de percorrer a Hash Table até à data de início dada, e retornar todos os Posts até à data de fim dada. Assim sendo, foi pensada uma estrutura que permitisse obter facilmente todos os Posts de determinados dias, um Array de GArrays, em que cada posição do Array principal corresponderia a um dia, e o GArray nessa posição conteria todos os ID's dos Posts desse dia. Devido ao diferente número de dias de cada mês, esta estrutura foi desenvolvida tendo como base todos os meses com 31 dias. Foram usados GArrays em vez de Arrays devido ao facto de, não sabendo o exato número de Posts de cada dia, e de mudar de dia para dia, não seria possível definir um Array com o tamanho correto, daí usar GArray, que vai aumentando o tamanho conforme seja necessário.

V. Resposta às Queries

A. Query 1

A resolução da query 1 consiste em procurar na HashTable dos Posts da estrutura principal o Post correspondente ao ID passado como argumento. Verificamos então se o Post é uma pergunta ou uma resposta, analisando o valor do

campo *postType*. Caso seja uma pergunta, é procurado o seu autor (User) na HashTable dos Profiles, e é retornado um par contendo o Título dessa pergunta e o nome do respetivo User. Caso seja uma resposta, é procurada a pergunta à qual está a responder, e procede-se da mesma forma anteriormente explicada.

B. Query 2

A query 2 consiste em criar uma estrutura auxiliar PAR (um par com o id de um utilizador e o número que Posts feitos na plataforma) e um GArray auxiliar que armazena todos os pares criados, percorrer a Hash Table dos Profiles toda filtrando a informação com uma função que retira de cada User o seu ID e o seu número total de Posts colocando na estrutura auxiliar PAR que será posteriormente inserido num array contendo todos os pares criados por esta função. O array será depois ordenado tendo em conta o número de posts de cada par. Para concluir a query, são inseridos na estrutura de retorno predefinida os N primeiros ID's do array.

C. Query 3

Na query 3 começa-se por fazer uma lista com os ID's de todos os Posts entre as datas passadas como argumentos da função. É em seguida percorrida a lista de ID's, analisando cada Post, verificando se este é uma pergunta ou uma resposta. Caso se verifique que é uma pergunta, a variável *questions* é incrementada, caso contrário, a variável *answers* é incrementada. No final é retornado um par (questions, answers).

D. Query 4

Primeiramente é criada uma lista (GArray) com os ID's de todos os posts entre as datas passadas como argumento da função. Percorre-se a lista de IDs, analisando o Post correspondente a cada um dos ID's, verificando se é uma pergunta e se essa pergunta contém a tag passada no argumento da função. Se sim, o ID desse Post é adicionado a uma nova lista chamada posts_with_tag. Quando já foi percorrida toda a lista dos ID's, a lista posts_with_tag é ordenada por ordem cronológica inversa. Depois de ordenada, é percorrida, adicionando cada ID à lista de retorno.

E. Query 5

A query 5 é resolvida através da procura de um User na Hash Table dos Profiles (Users) usando como chave o ID do User que se pretende encontrar. É depois procurado o seu "About Me", um campo da estrutura Profile, bem como um array que contém todos os ID's dos Posts que este utilizador publicou selecionando os 10 posts mais recentes. Finalmente é criado um User (estrutura de retorno) com o campo "About Me" e com um array contendo os ID's dos últimos 10 posts encontrados.

F. Query 6

Para a resolução da query 6, começa-se por criar um GArray com os ID's de todos os Posts entre as datas passadas como argumentos da função. Esse GArray de ID's é depois percorrido, analisando-se o Post correspondente a cada um dos ID's. Se o Post for uma resposta, o ID é adicionado ao GArray answers_post. Depois de percorrido todo o GArray dos ID's, é ordenado o GArray answers_post por ordem decrescente baseado no score de cada resposta. Por último são adicionados os N primeiros elementos do GArray answers_post à lista de retorno.

G. Query 7

O primeiro passo na resolução desta query consiste na criação de um GArray com os ID's de todos os Posts entre as datas passadas como argumento da função. Mais uma vez, é depois percorrido o GArray dos ID's, sendo analisado o Post correspondente a cada um dos ID's. Se for uma pergunta, adiciona-se o ID ao GArray questions_post. Depois de percorrido todo o GArray dos ID's, é ordenado o GArray questions_post em ordem decrescente baseado na quantidade de respostas de cada pergunta. Por último são adicionados os N primeiros elementos do GArray questions_post à lista de retorno.

H. Query 8

Na query 8, inicialmente é criado um GArray chamado keys, onde são adicionados todos os ID's de Posts que são perguntas. Este GArray é depois ordenado por ordem cronológica inversa. Depois de ordenado, é percorrido o GArray keys, analisando se a palavra passada como argumento está no título do post. Se sim, é adicionado o ID do Post à lista de retorno. Este procedimento é feito até serem inseridos N elementos naa lista de retorno, ou chegar ao final do GArray keys.

I. Query 9

A query 9 foi resolvida através da procura no array de Posts correspondentes a cada User, verificando o ID de cada post e caso este fosse uma pergunta, seria procurado no array do outro User em questão, alguma resposta cujo ParentID correspondesse à pergunta do primeiro User. Caso fosse uma resposta, verificar-se-ia se o outro User era autor de alguma pergunta cujo ID correspondesse ao parentID da resposta em questão. Existe ainda outra forma de ambos estarem relacionados no mesmo Post, que ocorre quando ambos os Users respondem a uma determinada pergunta, caso esse que é resolvido através da comparação dos ParentID's das respostas de cada um dos Users e, caso exista algum em comum, verifica-se que é mais um Post em comum. Após estas verificações, os ID's dos Posts em comum são ordenados por data.

J. Query 10

A query 10 é de simples resolução, reduzindo-se à procura do Post que queremos avaliar (na Hash Table dos Posts), sendo depois avaliada cada uma das suas respostas, recorrendo para isso ao array que contém as respostas àquela pergunta, e para cada uma calcula a sua pontuação, com a fórmula dada no enunciado do projeto. No fim é retornada a resposta que tem maior pontuação.

K. Query 11

Na query 11, primeiro é criado um GArray chamado best_users, onde são inseridos todos os Users. Depois o GArray é ordenado por ordem decrescente, baseado na reputação de cada User. Em seguida é também criado um GArray com os ID's de todos os Posts entre as datas fornecidas nos argumentos da função. Este GArray é percorrido depois, analisando o Post correspondente a cada ID. Se o Post for uma pergunta e for feita por um dos N primeiros Users do GArray best_users, é inserido no GArray questions_post. O próximo passo consiste em analisar cada Post do GArray questions_post, sendo criado um par para cada tag que o Post possuir (um par do género (ID, contagem)), sendo os pares em seguida inseridos no GArray tags_app. Ao analisar uma tag que já possua um par na lista, a sua contagem é incrementada. Por último, o GArray tags_app é ordenado por ordem decrescente, baseado na contagem de cada tag-par, e são inseridos na lista de retorno os N primeiros elementos.

VI. Conclusão

Concluindo, poderiam ser feitos melhoramentos, como colocar as perguntas e as respostas em Hash Tables separadas para melhorar o desempenho das queries que abordassem apenas respostas ou apenas perguntas. Colocar todos os Posts, perguntas e respostas, numa só Hash Table é melhor em termos de memória devido ao facto de ter apenas uma Hash Table, mas no tipo de querie mencionada, piora a *performance*. Nota-se também que na resolução da query 11, foi necessária a criação de uma estrutura auxiliar no ficheiro das Tags. Foi criada com 5 strings o que fez com que para processar as querys de um Post fosse necessário um ciclo para processar cada tag. Para otimizar este processo poderia alterar-se a estrutura para um array de strings e resolver o processamento das Tags de um Post apenas num só ciclo que iterava o array principal.