Trabalho Prático 2
Crawler
Recuperação de Informação

Daniel Ferreira Abadi[1]
2018088062

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

1. Introdução

O problema abordado consiste em construir ”Crawler”na linguagem de progrmação C++ utilizando a biblioteca CkSiper da Chilkat. O objetivo do mesmo é coletar 100.000 páginas da web brasileira para montar uma coleção para trabalhos posteriores.

O ”Crawler”proposto deve explorar as páginas de forma que possamos alcançar o número máximo de diferentes domínios possíveis, ou seja, uma busca em largura. O mesmo também deve respeitar o protocolo da Web, não podendo fazer mais de uma requisição em menos de 100ms para um mesmo servidor web, e ter duas estruturas para coordenar essa ação, um ”long-term”e um ”short-term scheduler”.

2. Implementação

Como dito anteriormente, o ”Crawler”deve possuir um escalonador de curso prazo e um escalonador de longo prazo. E, para simplificar o entendimento do relatório, a implementação será divida focando-se nesses dois tópicos.

2.1. Escalonador de longo prazo

Nas aulas nos foi passada a informação de que poderíamos utilizar uma fila de prioridades para realizar esse papel. Neste trabalho foi escolhido uma fila de prioridades utilizando a estrutura ”Pair”, ambos da biblioteca padrão do C++. O ”Pair”consiste de um inteiro, que seria o tamanho da página, e uma ”string”, sendo a URL.

A fila acima é ordenada de acordo com o inteiro, que é calculado contando-se o número de ”.”e ”/”, como nos foi sugerido. Quanto mais desses caracteres temos, menos importante o endereço web é. Para uma simplificação do código, foi decido que apenas pegaríamos esse número e multiplicaríamos por -1, já que a ordenação é feita em ordem decrescente.

2.2. Escalonador de curto prazo

O ”short-term scheduler”nos foi apresentado como um conjunto de ”threads”, com o intuito de podermos recolher páginas de diferentes servidores web ao mesmo tempo. Para isso foi criada uma grande função chamada ”Crawler”, que inicialmente apenas cria as variáveis do Chilkat e define as preferências do mesmo.

Como nos foi passada a informação de que poderíamos coletar apenas páginas brasileiras, foi decidido que apenas coletaríamos páginas que tivessem ”.br”utilizando

uma função do próprio Chilkat. Apesar de não ser a melhor estratégia, pois vários sites brasileiros não tem esse padrão, foi a que melhor funcionou.

Devido a condição de cumprir com o protocolo web, o escalonador de curto prazo deve verificar se o domínio da URL que está pegando da fila de prioridades já está sendo usado naquele momento. Para isso foi criado um ”Set”de ”strings”que é atualizado toda vez que uma ”thread”pega um link, adicionando o seu domínio ao ”Set”, e quando o processo de ”crawling”acaba naquele site, o domínio do mesmo é retirado do ”set”. Quando uma ”thread”pega uma URL cujo domínio já está sendo usado, essa URL é guardada em um ”Vector”auxiliar e a ”thread”requisita um novo link para o ”long-term scheduler”. Esse processo se repete até que seja encontrado um domínio que não está sendo usado, e caso se esgotem os itens da fila de prioridade, a ”thread”que está fazendo a requisição é, então, colocada em espera por 2 segundos antes de poder realizar uma nova requisição, podendo ser colocada em espera novamente caso ainda não tenham novos links. Quando conseguir um novo domínio, todos os anteriores guardados no ”Vector”auxiliar são colocados de volta na fila.

Recebido o link para a página, a primeira é baixada e pegam-se as URL's que apontam para outros domínios e colocam-se na fila de prioridade. É capturado também o número de páginas que o site aponta, tal dado serve para coletarmos apenas as páginas de nível 1. Para cada página baixada pelo método ”CrawNext”é verificado se a sua URL já foi utilizada por meio de um ”Set”que armazena todos links que já foram utilizados. E, para cada página nível 1, são recolhidos os seus links que apontam para outros domínios e colocados no escalonador de longo prazo.

Durante a produção do trabalho foi notado que algumas URL's que tinham um formato do tipo ”http:///...”geravam um erro no qual fazia o Chilkat lançar uma exceção de ”std::logic error”, e para tratar tal erro foi implementado apenas um simples ”try-catch”em dois pontos do código, um para pegar os endereços para outros domínios e outro para pegar o domínio de páginas provenientes do escalonador de longo prazo.

2.3. Detalhes adicionais

Nesta seção serão apresentados detalhes adicionais que não interferem diretamente na arquitetura do ”Crawler”, mas que são importantes.

O hardware utilizado para executar o programa não é muito potente, portanto foi priorizado ao máximo a quantidade de código paralelizável e o número de ”threads”foi apenas 0. Portanto, para cada ”thread”foram abertos quatro arquivos de texto para armazenar as URL's, os HTML's, os tempos de requisições e o tamanho das URL's.

Para coletar os dados pedidos na especificação, como tempo de resposta, foi utilizada a biblioteca Chrono do C++, começando a contagem antes de chamar o método ”CrawlNext”e terminando logo após, para todas as URL's. Esse dado é então colocado em um arquivo de texto. O mesmo vale para o dado adicional sobre o tamanho de cada URL. Já para o número de páginas níveis 1, toda URL na qual começa o processo o número de links para que ela aponta é salvo no mesmo arquivo que guardam os endereços web.

Uma curiosidade é que enquanto realizava-se testes foi percebido que servidores de URL's de domínios estrangeiros normalmente possuem um tempo de resposta mais

rápido do que os domínios brasileiros, com grandes diferenças, a longo prazo, em tempo de execução.

3. Análise e apresentação de dados

Os dados que irão ser utilizados aqui foram coletados como descreve a seção anterior. Foram tratados utilizando-se a linguagem de programação Python e suas bibliotecas de visualizações de dados, como Pandas e Matplotlib.

	URL's sizes
count	20000.000000
mean	4.025150
std	2.994439
min	2.000000
25%	2.000000
50%	3.000000
75%	5.000000
max	28.000000

Figura 1. Dados sobre os tamanhos das URL's

A Figura 1 nos mostra alguns dados sobre os tamanhos dos links em uma amostra de 20 mil. Temos a quantidade, a média, o desvio padrão, o valor mínimo, os 3 quartis e, por fim, o valor máximo. Lembrando que os tamanhos foram calculados contando-se o número de "."e "/". Podemos observar que grande parte dos endereços web coletados são relativamente pequenos o que pode sugerir que realmente foi seguida a propriedade de busca em largura.

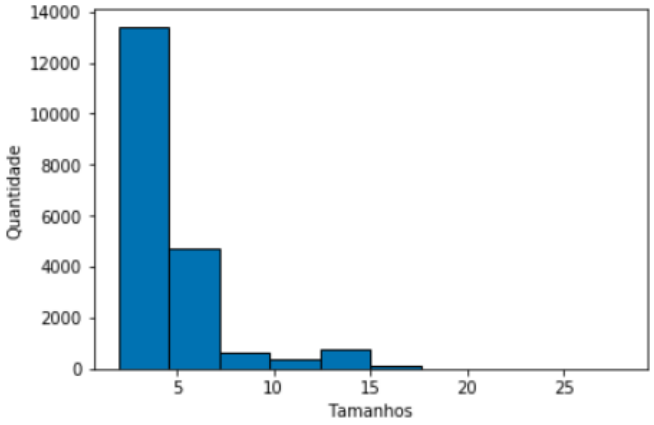

Figura 2. Tamanhos das URL's

A Figura 2 nos apresenta um histograma gerado a partir da amostra citada acima, corroborando os dados encontrados.

	Times
count	2000.000000
mean	810.270500
std	901.344427
min	31.000000
25%	275.750000
50%	526.000000
75%	1007.500000
max	9817.000000

Figura 3. Tempos (em milissegundos)

A Figura 3 nos mostra os tempos das requisições em milissegundos de uma subamostra de 2 mil tempos, tendo os mesmos dados da Figura 1. O tempo máximo apresentado nos mostra que a forma que o Chilkat tem de limitar o tempo de conexão nem sempre funciona como deveria.

Não foi possível pegar o tamanho exato de cada página HTML para uma análise de seus tamanhos nem como para o programa listar o tamanho médio de cada. Porém, para não ficar sem o dado em questão, foi feita uma média do tamanho de todos os arquivos que contém os códigos fonte das páginas, sendo 379 KB. Isso nos mostra que as páginas web em geral são de porte médio, levando em consideração grandes páginas de até 1 MB.

4. Conclusão

Como o professor disse, esse trabalho foi realmente bem envolvente, sendo desafiador em vários aspectos. Foi feito um escalonador de longo prazo utilizando uma fila de prioridades, que é ordenada pelo tamanho das URL's, que distribuí os links para os escalonadores de curto prazo, que só aceitam tal endereço web se o domínio em questão não está sendo utilizado. Tais escalonadores de curto prazo são, na verdade, "threads"que contém um número limite de 100 mil para fazer o processo de "Crawling".

A experiência com a biblioteca Chilkat no geral foi boa, com exceção de alguns erros sem explicações, como o programa parar de executar sem apresentar qualquer erro. Tais erros ocorreram em vários momentos do processo de desenvolvimento, bem como no final para a coleta das 100 mil páginas quando o mesmo já estava na página 90 mil, 60 mil, 50 mil e etc. O fato de não termos acesso aos detalhes de implementação comprometeu a busca por tais erros, mas nada que impedisse o progresso.

E como o professor nos propôs, essa foi realmente uma jornada de aprendizado, ganhamos a experiência de trabalhar com uma nova biblioteca e descobrir como usá-la melhor, entre outras coisas. Para melhorias, com essa nova vivência, poderia-se modificar a estrutura geral do código, para algo mais bem estruturado e organizado, utilizando de conceitos como classes e uma maior distribuição de responsabilidades em funções.