Recuperação de Informação
Máquinas de Busca na Web
Trabalho Prático 1
Crawler

Érico Marco D. A. Pereira[1]
2013007293

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

{emarco.pereira}@dcc.ufmg.br

Resumo. *Trabalho Prático 1 da disciplina de Recuperação de Informação - Máquinas de Busca na Web - DCC923 1º/2017 cujo objetivo é desenvolver e implementar um Crawler para a Web brasileira.*

1. Introdução

Um ***Web Crawler*** é uma ferramenta que realiza a coleta de páginas da Web com o intuito de coletar dados a serem utilizados para fins posteriores, como, por exemplo a análise relativa ao conteúdo desses dados ou uso aplicações (como buscadores Web) que operam sobre esse tipo de dado. Tipicamente, um *Crawler* caminha na Web realizando o processamento do conteúdo de uma página e, a partir dos *links* nela contidos e dos critérios de coleta estabelecidos, define as próximas a serem visitadas ou coletadas.

Os *Crawlers* existentes podem ser categorizados em dois tipos principais: os que realizam **coleta vertical**, que tem por objetivo a coleta de páginas cujo conteúdo é relativo a um ou mais assuntos ou *sites*/domínios específicos, e os que realizam **coleta horizontal**, que objetiva a abrangência da maior variedade de assuntos ou maior número de *sites*/domínios possíveis. Neste projeto será proposto um *Crawler* que realiza uma coleta horizontal objetivando a abrangência da maior quantidade possível de páginas de domínios diferentes.

Dentre os requisitos a serem seguidos pelo *Crawler* proposto há ainda o respeito às regras de etiqueta que determinam a concordância com as especificações do arquivo *robots.txt*, presente em cada servidor da web, e a espera de um intervalo de 30 segundos entre requisições a um mesmo domínio.

A Seção 2 apresenta uma descrição da arquitetura do *Crawler* proposto e das demais decisões de projeto. A Seção 3 apresenta uma breve análise da complexidade do programa. A Seção 4 apresenta uma análise experimental de performance e impacto de parâmetros. Possíveis melhorias são descritas na Seção 5. E a Seção 6 concluí o trabalho.

2. Metodologia

Sabe-se que quanto menor a extensão de um determinado ***link*** ou ***url*** em número de subdomínios e diretórios internos maior a probabilidade desta *url* ser a raiz de seu domínio, uma vez que esta possui a *url* de menor extensão por possui apenas os níveis do domínio

e nenhum diretório interno, e ser uma página relevante à cobertura de conteúdo, uma vez que as páginas próximas a raiz comumente apresentam uma amostra representativa do conteúdo presente naquele *site*/domínio. Sendo assim, adotando uma estratégia de coleta que priorize páginas de *urls* de menor extensão tende a satisfazer o critério de abrangência de domínios (coleta horizontal) e ainda a proporcionar uma melhor qualidade do conjunto de páginas coletado em termo de relevância do conteúdo. O *Crawler* descrito a seguir adota essa estratégia.

2.1. Arquitetura do *Crawler*

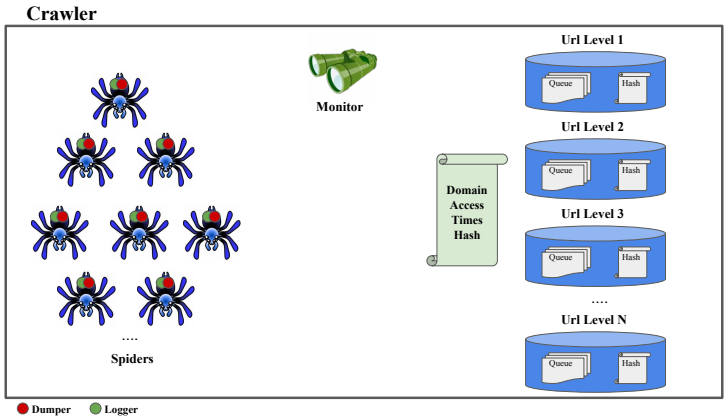

Figura 1. Arquitetura do *Crawler* proposto.

As principais estruturas do *Crawler* proposto que tem por objetivo a gestão das *urls* estão agrupadas em níveis, sendo que cada nível é responsável pelas *urls* de uma extensão especifica. Internamente, cada nível possuí uma fila de *urls* e uma tabela *hash* para controle das urls já adicionadas na fila. Há ainda uma tabela hash, global aos níveis, responsável pelo armazenamento do momento do último acesso a cada domínio já visitado para que seja possível a verificação do critério de etiqueta de 30s.

As principais entidades ativas do *Crawler* são chamadas ***Spiders*** e são responsáveis pela coleta e processamento das páginas além da manipulação das estruturas destinadas a gestão de *urls*. Cada *Spider* executa em sua própria *thread* e utiliza duas entidades auxiliares: um ***Dumper***, responsável pelo armazenamento das páginas coletadas em disco, e um ***Logger***, que realiza o registro de mensagens de *log* referentes a atividade da respectiva *Spider*. Há ainda uma outra entidade, ***Monitor***, que realiza a coleta periódica de medidas estatísticas, como número de páginas coletadas, de todos os *Spiders*.

Cada *Spider* realiza três atividades principais: **coleta de página**, **aquisição de urls**, e **transferência de urls**. A primeira consiste em fazer o *download* da página de uma determinada *url*, extrair as *urls* nela contidas e repassa-la ao *Dumper*. A segunda consiste em ir aos níveis de *urls* e retirar as *urls* cujos domínios podem ser acessados naquele momento até que encha o *buffer* interno destinado a esse propósito, ***outBuffer***. E a terceira consiste em ir em cada um dos níveis de *urls* para adicionar nas filas todas as *urls* contidas no *buffer* interno de *urls* extraídas, ***inBuffer***, que ainda não tenham sido adicionadas anteriormente.

Durante a execução, cada Spider coletará páginas enquanto houverem *urls* em *outBuffer*. Se em algum momento o *outBuffer* estiver vazio, a aquisição de páginas é feita, ou se *inBuffer* estiver além de seu limite de capacidade, a transferência de *urls* é realizada.

O fluxo de execução do programa acontece da seguinte forma: primeiro um *Spider* é criado apenas para coletar as páginas das *seed urls* e adicionar uma certa quantidade de *urls* nas filas dos níveis; depois vários *Spiders* são lançados concorrentemente e executam a sequência de atividades previamente descrita até que o comando de parada seja dado.

2.2. Demais Decisões de Projeto

- Internamente às entidades *Spider*, para coleta e processamento das páginas, foi utilizada a biblioteca **Chilkat v9.5.0.66** [2]. Esta biblioteca segue as especificações dos arquivos *robots.txt*, garantindo o respeito a esta regra de etiqueta.
- Com objetivo de seguir a estratégia de priorizar as *urls* de menor extensão, tanto as atividades de aquisição quanto de transferência de *urls* seguem a ordem crescente na visita aos níveis de **url** garantindo assim que as *urls* de menor extensão sejam esgotadas antes de adquirir de uma extensão superior.
- Como o *Crawler* objetiva a coleta de páginas da Web brasileira, são mantidas as *urls* apenas de domínio .br de modo que as outras são prontamente descartadas.
- Apenas são consideradas *urls* de extensão até 10 (subdomínios + diretórios), sendo este também o número de níveis de *urls* na arquitetura. *Urls* maiores são descartadas.
- *Urls* dinâmicas, como e-mail por exemplo, também são descartadas.

3. Análise de Complexidade

Considerando:

- $urls$ como o número de urls coletadas;
- $urls\_por\_pagina$ como o número médio de urls por página;
- $html$ como uma página *html* média;
- url como um *link* ou *url* média;

Desconsiderando-se o paralelismo, para cada página que é coletada pelos *Spiders* há um custo de $O(\|html\| + urls\_por\_pagina * \|url\|)$ para processamento de cada *url* e adição no *buffer* interno. Para cada transferência de *urls* há um custo de O(urls), uma vez que cada *url* pode ser inserida nas filas em $O(1)$.

Para cada aquisição de *urls* há o custo de procurar *urls* cujas páginas podem ser coletadas naquele momento o que pode requisitar percorrer toda a fila no pior caso. Aproximando o custo de percorrer a fila como $O(urls)$, uma vez que seu tamanho costuma atingir uma ordem de grandeza semelhante ao do número de *urls* coletadas então, no pior caso, há um custo de $O(urls)$ para coletar cada *url*. Porém, no caso médio é necessário verificar apenas uma quantidade constante K de *urls* até encontrar uma cujo domínio possa ser acessado naquele momento, uma vez que um domínio se torna acessível a partir de um fator de tempo contante (30s). Portanto o custo de se adquirir uma *url* é $O(K)$.

Considerando que toda *url* foi processada e adicionada ao *buffer* interno $O(1 + \|url\|)$, adicionada a fila $O(1)$, adquirida depois por um *Spider* $O(K)$ e teve seu *html* baixado e processado $O(\|html\|)$, então o custo por url já extraída e baixada é $O(\|url\| + K + \|html\|)$ que gera um custo de $urls * O(\|url\| + K + \|html\|)$.

4. Análise Experimental

Os experimentos de coleta de páginas relatados a seguir foram realizados utilizando uma máquina Linux Ubuntu 14.04 LTS com 16 cores, 16GB de RAM e memória do tipo SSD. Essa máquina está conectada a um link de internet compartilhada com outras máquinas cuja banda é de aproximadamente 100Mbps.

4.1. Análise de Desempenho

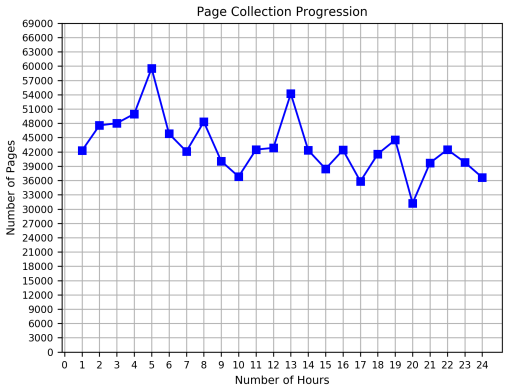

Figura 2. Progressão do desempenho na coleta de páginas

Com o intuito de analisar a progressão do desempenho de coleta durante um período um período relativamente longo, foi realizado um experimento com um tempo de execução de 24 horas aproximadas. Neste período foi contabilizado um total aproximado de 1.036.969 páginas coletadas.

Observando a Figura 2 é possível concluir que o *Crawler* mantém um desempenho não muito distoante durante toda a coleta apresentando alguns picos na primeira metade da coleta. Possivelmente, isso se deve à configuração de uma situação favorável onde grande parte dos *Spiders* possuí paginas a coletar em seus *buffers* internos ou uma grande quantidade de páginas de novos domínios aparece no início da fila, ou ainda, devido a um aumento da banda real disponível consequente a um favorecimento do tráfego na rede.

A segunda metade da coleta apresenta um suave declínio de desempenho com a maioria das menores estimativas. Esse aspecto pode ser explicado por um crescimento do tamanho das estruturas de *urls* internas fazendo com que suas operações se tornem mais custosas. Além disso, um ou outro vale pode ter fundamento na ocorrência de alguma instabilidade na rede ou no uso demasiado de disco, uma vez que não está sendo realizado um *buffering* interno de escrita de arquivo.

4.2. Análise de Parâmetros

Objetivando a observação do impacto do número de *threads* utilizado durante a coleta foi realizado um experimento no qual o *Crawler* foi executado sucessivamente com diferentes configurações desse parâmetro durante intervalos de 3 horas.

Observando o gráfico da Figura 3 é possível verificar um aumento de performance mais brusco até 20 *threads* indicando uma provável subutilização de recursos. A partir daí, o suave aclive até o valor de 80 *threads* indica que possivelmente está havendo um

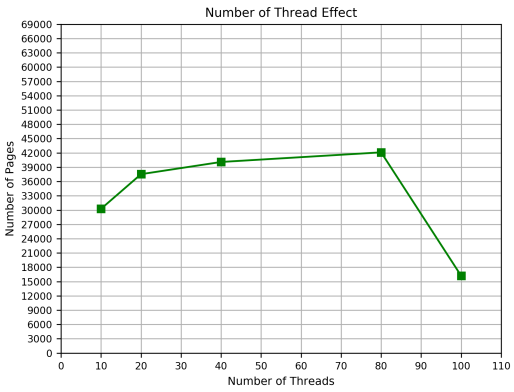

Figura 3. Experimento de coleta de páginas com varação do número de threads.

grande volume de concorrência que não permite um grande impacto causado por mais *Spiders* a coletarem simultaneamente e portanto, muitas *threads* devem estar em espera ao mesmo tempo. Há ainda uma chance de esgotamento da banda real de rede disponível.

A significativa diferença de desempenho observada no valor 100 provavelmente deve-se à um aumento da concorrência e um consequente travamento das atividades de aquisição e transferência de *urls*. Ou ainda, com menor probabilidade, devido à um período de maior tráfego na rede uma vez que o respectivo experimento ocorreu em um horário comercial e posterior aos outros que iniciaram durante a madrugada, que tipicamente apresenta menor tráfego.

5. Possíveis Melhorias

Dadas as possíveis causas das reduções de eficiência durante os experimentos realizados, são propostas as seguintes melhorias:

- Com o objetivo de reduzir a concorrência talvez seja necessário um outra arquitetura na qual as *threads* não realizassem operações tão demoradas em estruturas compartilhadas. Possivelmente uma melhor solução seria uma arquitetura com uma fila de domínios e filas de *urls* individuais por domínio que retirasse permitisse que filas de domínios diferentes sejam acessadas simultaneamente exigiria concorrência apenas na busca por domínios na fila que podem ser acessados naquele momento. Porém uma fila de domínios seria consideravelmente menor do a de *urls* e não teria itens de domínios diferentes.
- A realização de *buffering* interno poderia favorecer a não ocorrência das quedas de desempenho ocasionadas por alta atividade de escrita em disco proporcionando um uso mais controlado deste recurso.
- O *Crawler* proposto tem por objetivo a coleta de páginas da Web brasileira porém, devido ao critério de *urls* domínios terminados em .br, acaba por descartar páginas também brasileiras que não atendem a este critério. Uma possível melhoria seria a verificação de idioma das páginas a fim de coletar todas as páginas em português do Brasil.
- A grande quantidade de páginas coletadas acaba por ocupar uma grande quantidade de espaço de armazenamento. Uma possível solução para amenizar isso

seria a realização da compressão das páginas, uma vez que compressão de texto tipicamente apresenta um significativo fator de compressão.

6. Conclusão

Este trabalho teve por objetivo a elaboração, desenvolvimento e implementação de um *Web Crawler* que realiza a coleta horizontal de páginas da Web brasileira. Para fazer isso, a arquitetura proposta prioriza a coletas de páginas de *urls* de pequena extensão, obtendo assim cobertura e qualidade de coleta. Em termos de performance, é capaz de coletar um conjunto significativo de páginas em tempo razoável, porém melhorias, como as sugeridas, podem ser feitas.

7. Referências

[1] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. [2] https://www.chilkatsoft.com/chilkatLinux.asp.