CONSTRUÇÃO DE UM GRAFO WEB MASSIVO REAL

Aluno: Lucas Nissenbaum
Orientador: Eduardo Sany Laber

Introdução

Um *grafo web* representa a estrutura da web, de modo que cada página web é um vértice e cada link entre duas páginas corresponde a uma aresta [1]. Para construir um *grafo web* precisamos saber quais páginas nos direcionam para quais, portanto, precisamos coletar as páginas e extrair os seus links. Isto pode ser feito através de um *crawler*, que é a ferramenta utilizada para a coleta automatizada e seletiva de sites [2]. O objetivo desse trabalho é construir um *crawler* que faça a coleta de páginas web de modo que seus dados possam ser utilizados para construir um *grafo web*. O *crawler* deve ser capaz de realizar o download cerca de cem milhões de páginas.

Metodologia

A arquitetura do *crawler* é dividida em quatro componentes principais: *robot*, *storage*, *spider* e *filter* (Figura 1). Cada um destes componentes é executado por *threads* distintos e que se comunicam através de listas circulares (paradigma produtor-consumidor). A seguir, descrevemos em mais detalhes o funcionamento de cada componente.

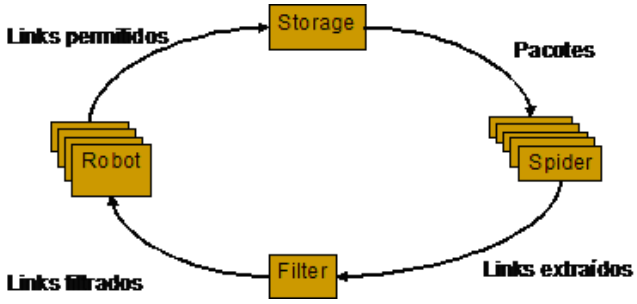

Figura 1: Arquitetura do *crawler*

O componente *robot* verifica se os links extraídos são permitidos para serem coletados. Isso é definido pelo proprietário do site por meio do protocolo *robots.txt* [3]. Para isso, ele utiliza um banco de dados para guardar as páginas já utilizadas e assim poder verificar rapidamente se o site pode ser visitado ou não. Este componente armazena o arquivo *robots.txt*, e o IP do site. Para esta etapa, estão sendo utilizadas vários *threads* (cerca de 128).

O componente *storage* é utilizado para adicionar os links permitidos ao banco de dados e gerar pacotes de links de mesmo host. Este banco de dados junta todos os links extraídos e que ainda não tiveram seu download feito. Sua saída é um pacote de sites, apenas com sites do mesmo host. Essa saída é retornada em forma de pacotes, pois, em primeiro lugar, o download ocorre de forma mais rápida para sites do mesmo host, pois o *crawler* se conecta com este host e então pega todos os sites, por haver localidade de referência. Em segundo lugar, na compressão, como a formatação de sites de mesmo host é igual, não há a duplicação desta formatação.

O componente *spider* recebe um pacote de links e faz o *download* de todos os links deste pacote, selecionando apenas aqueles de formato *text/html*. Em seguida, é feita a extração

dos links das páginas coletadas e a compressão das mesmas utilizando o método zip [4]. Para este componente, estamos usando diversos *threads* (cerca de 512).

O componente *filter* recebe as páginas coletadas e adiciona-as ao banco de dados. Porém, sua principal função é exercer a filtragem dos links extraídos destas páginas. Em [5], observou-se que cerca de 11% dos links extraídos eram únicos e apenas 1.6% do total foi coletado. Para o problema de verificar a unicidade dos links extraídos (*urlseen*), o IRLbot [5] utilizou estruturas de dados de disco complexas (*hash* com *bucket sort*) para garantir uma alta taxa de coleta (1.789 páginas/s). Neste trabalho, propomos uma solução mais simples que consiste em filtrar cerca de L_f % dos links extraídos de forma aleatória. O valor de L_f é $100L_c/(L_c+L_e)$, onde L_e é o número de links extraídos em todas as páginas coletadas pelo *crawler* e L_c é o número total de páginas coletadas pelo *crawler*. Todo processamento de filtragem é feito em memória RAM. Isto nos permitiu utilizar o banco de dados Berkeley [6] sem a necessidade de construir estruturas de dados de disco complexas para garantir uma alta taxa de coleta (em torno de 300 páginas/s) tendo em vista os recursos computacionais disponíveis.

Resultados experimentais parciais

O *crawler* foi desenvolvido utilizando a linguagem de programação C++. Para realizar os downloads, foi utilizada a biblioteca libcurl 7.17.0 [6]. O banco de dados utlizado foi o Berkeley DB na versão 5.0 [7]. Para a interface gráfica, foi usado o Qt 4.7.0 [8].

Os testes foram executados em uma máquina Quad Core 2 de 2.6GHz, 4GB de RAM, com disco rígido SATA de 320GB, placa de rede 10/100 Mbps, com sistema operacional Linux Ubuntu 10.04 Kernel 2.6. A rede de acesso a Web é compartilhada com todo o departamento de informática da PUC e, portanto, tivemos que limitar a taxa de *download* em 4Mbit/s.

Realizamos um teste de deixar o *crawler* rodando por aproximadamente dez minutos. Nosso resultado foi uma taxa de *download* de por volta de 300 páginas por segundo, totalizando 18000 páginas. Entretanto, ainda serão realizados testes e outras medições, com um maior intervalo para verificar a escalabilidade do *crawler*.

Conclusões

Para mim, essa pesquisa foi extremamente válida. Além de utilizar uma estrutura relativamente simples para um *crawler*, trabalhar diversos conceitos vistos em sala de aula, como banco de dados, programação orientada a objetos, sistemas operacionais e algoritmos. Além disso, também tive uma visão de um grande projeto de computação, um outro ponto de vista em relação ao de sala de aula.

Para trabalhos futuros, teremos vários trabalhos em cima do grafo web resultante, entre eles, a compressão de grafos web e a verificação de propriedades demonstradas neste tipo de grafo.

Referências

1- GUILLAUME, J; LATAPY, M. The Web Graph: An Overview. *AlgoTel*, Paris, 2000.
2- OLSTON, C.; NAJORK, M. Web Crawling. *Foundations and Trends in Information Retrievals*, California, v. 4, n. 3, p. 175-243, 2010.
3- ROBOTSTXT Implementação do padrão robots.txt. Disponível em: http://robotstxt.org. Acesso em: 22 de jan. 2010.
4- ZLIB Biblioteca de compressão Zlib. Disponível em: www.zlib.net. Acesso em: 30 de jun. de 2010.
5- LEE, H.T.; LEONARD, D.; WANG, X; LOGUINOV, D. IRLbot: Scaling to 6 billion pages and beyond. *World Wide Web Internet And Web Information Systems*, v. 3, n. 3, Texas, 2009.
6- HAXX. libcurl. Disponível em: http://curl.haxx.se/libcurl/. Acesso em: 29 de jun. 2010.
7- ORACLE. Banco de dados Berkeley. Disponível em: http://www.oracle.com/technology/software/products/berkeley-db/index.html; Acesso em: 29 de jun. 2010.
8- NOKIA. Qt. Disponível em http://qt.nokia.com/products/. Acesso em 29 de jun. 2010.