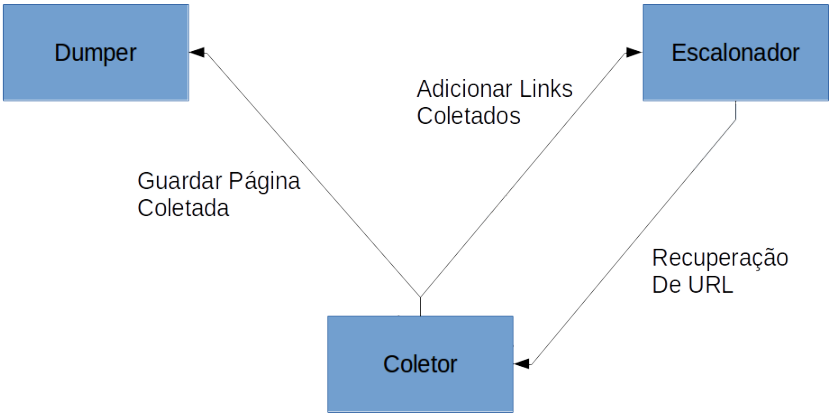
Trabalho Prático I - Coletor

Thiago Vieira de Alcantara Silva
2012075627

1 Descrição do coletor

Neste trabalho prático, desenvolvi um coletor para coletar páginas na web brasileira. A arquitetura do coletor foi divida principalmente em três componentes, coletor, escalonador e dumper.



O coletor, basicamente, envia requisições para servidores web pedindo por links específicos, os htmls retornados são enviados para um dumper juntamente com os links das URLs coletadas. Neste projeto temos 80 coletores trabalhando paralelamente, cada coletor tem o seu dumper.
O dumper acumula as páginas coletadas em memória principal e a cada 50 páginas coletadas um flush é executado, escrevendo os dados acumulados em memória secundária; para cada dumper temos um arquivo em que é aberto no início da coleta e somente é fechado ao término.
O escalonador guarda as URLs a serem coletadas e as ordena de maneira que o coletor fique mais rápido e colete páginas relevantes.
Para fazer o parsing dos códigos html e para fazer as requisições aos servidores web, respeitando seus robots, a biblioteca chilkat foi usada.

2 Decisões de Projeto

2.1 Paralelismo

Neste projeto, o coletor trabalha com 80 threads. Em cada thread temos um objeto CkSpider e um dumper; assim cada thread fica em um ciclo, recuperando url do escalonador, coletando a url recuperada e enviando o html coletado para o dumper salvar em memória secundária.
A escolha do número de threads foi feita empíricamente. Foram feitos testes com 50, 80 e 100 threads. Com 50 threads a banda não era utilizada constatemente, e com 100 threads nao obtive speedups satisfatórios, assim foi escolhido utilizar 80 threads.

2.2 Escalonador

O escalonador neste trabalho, guarda URLs a serem coletadas e as ordena para otimizar o coletor e aumentar a qualidade do conjunto de páginas coletadas.

2.2.1 Binary Heap

Optei por implementar uma binary heap e não usar a priority queue da stl pois assim tenho mais controle da memória gasta pelo coletor. A heap é inicializada com o número máximo de urls que podemos guardar em memória principal, assim esta heap tem tamanho estático e garanto que não usarei mais do que requerido, ao invés da priority queue que usando o vector dobra de tamanho a cada vez que o limite de tamanho é atingido.

2.2.2 Ordenação das URLs

Para ordernar as URLs, tenho uma função hash $h(domain)$ cujo retorno varia de 0 a 256. A entrada desta função hash é o domínio da url a ser adicionada. Tenho também um conjunto de contadores m inicializados como zero, a cada url adicionada ao escalonador, $m[h(domain)]$ é somado um.

A fórmula final usada para ordenar os valores da heap fica: $f(url) = 100 * m[h(domain)] + nComponentes$ onde domain é o domínio e nComponentes é o número de componentes da url. As urls com os menores pesos são priorizadas, assim esta função ajuda a aumentar a velocidade do coletor, evitando fazer mais de uma requisição a um mesmo servidor em um curto espaço de tempo.

3 Análise de Complexidade

O coletor foi desenvolvido com o objetivo de coletar o maior número de páginas por segundo. Todas as funções do dumper são executadas em $O(1)$ e um flush do buffer de saída é executado a cada 50 páginas coletadas.

No escalonador, as funções de recuperar a próxima url e retirá-la são executadas em $O(1)$ e $O(log\ n)$ respectivamente, sendo n o número de URLs presentes no escalonador. Na função para adicionar uma URL checamos se a url tem algum termo proibido, verificamos se URL não é dinâmica, após isso ela é adicionada ao escalonador; assim a complexidade desta função fica como $O(|url| + log\ n)$, o tamanho da url mais o logarítmo do número de URLs já presentes no escalonador.
O coletor, a cada loop, executa a requisição ao servidor web, faz o parsing do html e adiciona novos links ao escalonador; assim a complexidade de cada loop do coletor fica como $O(|html| + nLinks * (|url| + log\, n))$, o tamanho do html coletado, mais o número de links recuperados vezes a complexidade de adicionar cada link no escalonador.

4 Resultados Experimentais

4.1 Utilização de Banda

A coleta das páginas foi executada em uma internet residencial de 10Mb/seg. Em um dos testes, foram coletadas 1005803 páginas em 22h, totalizando 107 GB de páginas coletadas. 114.890.375.168 Bytes $\Rightarrow$ 1.450.636 Bytes/seg equivalendo a 1,4 MBytes/seg ou 12,6 páginas por segundo.

5 Pontos Fortes e Fracos do Coletor

Este coletor faz uma coleta bem horizontal pela internet brasileira, principalmente pois demos preferência por domínios não coletados e após isso por URLs pequenas. Assim temos uma diversidade grande de domínios e conteúdos coletados.
Os pontos fracos deste coletor são a falta de compressão das páginas coletadas para gastar menos memória primária e secundária, e a falta de um algoritmo que opere em memória secundária para guardar e verificar as URLs coletadas para evitar páginas duplicadas.