PONTÍFICIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA

Kaio Henrique Andrade

**Tópicos em Computação I – Recuperação de Informação**

1

Kaio Henrique Andrade

**Tópicos em Computação I – Recuperação de Informação**

Relatório técnico apresentado como requisito parcial para obtenção de aprovação na disciplina Tópicos Em Computação I, no Curso de Ciência da Computação, na Pontíficia Universidade Católica de Minas Gerais.

Prof. Dr. Wladmir CardosoBrandão

**RESUMO**

Este trabalho apresenta as características da implementação do componente Coletor de uma Máquina de Busca para Web. Questões relacionadas a bibliotecas utilizadas, ambiente de desenvolvimento, características nos resultados são abordadas neste documento. Todas as referências serão disponíbilizadas para fins de estudo. Além disso, o trabalho descreve todos os passos para instalação e execução do coletor.

Palavras-chave: Web-Crawler, Coletor Web

2

3

**SUMÁRIO**

**1 INTRODUÇÃO.............................................................................................. 4**

**2 DESENVOLVIMENTO.................................................................................. 5**

**3 AMBIENTE DE DESENVOLVIMENTO ....................................................... 7**

**3.1 INSTALAÇÃO E CONFIGURAÇÃO**

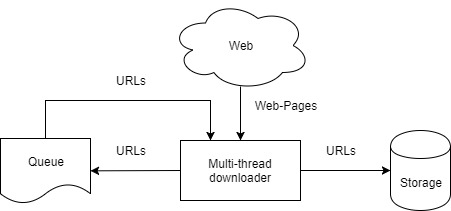
**4 RESULTADOS**

**1 INTRODUÇÃO**

O objetivo deste relatório é descrever o ambiente de desenvolvimento utilizado na implementação do coletor. Bem como descrever asw ferramentas utilizadas e disponibilizar as devidas referências para fins de estudo. Como é sabido o principal objetivo de um coletor é coletar a maior quantidade de páginas possível na web. Para isto é necessário bom desempenho.

**2 DESENVOLVIMENTO**

O coletor foi desenvolvido em ambiente linux com a distribuição Ubuntu 16.04 LTS, utilizando a linguagem de programação Python em sua versão 3.6.0. O hardware consiste em um processador Intel(R) Core i5-2310 @ 2.90 GHz, com 8 Giga Bytes de memória RAM e um disco rígido de 1 Tera Byte. Python possui diversas bibliotecas para tratamento de requisições HTTP e Multithreading, questões importantes desta primeira fase do trabalho. Para tratamento de conexões Web fora utilizada a bilbioteca nativa da linguagem **urllib** em sua versão 21.5**.** A biblioteca dispõe de quatro módulos essenciais para o simplificação e desenvolvimento deste trabalho. Os módulos **urllib.request, urllib.parse, urllib.robotparser** foram utilizados para tratamento de URLs e resolução de “Robots.txt”. Para o parsing do html a bilbioteca **html.parser**, em sua versão 20.2, foi utilizada.



A implementação do coletor consiste em realizar uma requisição HTTP a um conjunto de URLs predefinidas, chamadas “seeds”, solicitando o arquivo Robots.txt do servidor do domínio da URL. Caso a URL em questão faça parte do conjunto de URLs permitidas para aquele domínio, todo o conteúdo todo o conteúdo html disponível na página é coletado e em seguida processado para extração de possíveis novas URLs referênciadas no site. O coletor mantém duas estruturas principais para seu funcionamento, um conjunto de URLs visitadas e um conjunto de URLs não visitadas. Estes conjuntos são utilizados durante todo o funcionamento do coletor e através de operações de conjuntos é possível identificar se uma URL foi ou não coletada anteriormente.

Um conjunto arbitrário de threads pode ser definido na estrutura principal do programa (“engine.py”). As threads são gerênciadas pela bilbioteca padrão da lingugem **threading.** Problemas de concorrências e regiões críticas são gerênciadas pela biblioteca através de chamadas de funções especificas dentro do programa. A quantidade de threads executando simultâneamente precisa ser definida de acordo com o ambiente de execução, sendo necessário alterar o valor da constante “MAX\_THREADS ”. Testes foram realizados com x, y, z threads e os resultados são descritos na sessão 2.X.

**3** **AMBIENTE DE DESENVOLVIMENTO**

Todo o coletor foi desenvolvido em ambiente Linux com a distribuição Ubuntu 16.04 LTS disponível para download [aqui](http://releases.ubuntu.com/16.04/). Todos os testes foram executados em um PC Desktop com processador Intel(R) Core i5-2310 @ 2.90 GHz com 4 núcelos físicos, 8 GB de memória RAM e 1 TB de armazenamento em disco. A linguagem de programação Python foi utilizada no desenvolvimento do trabalho. Por possuir fácil prototipação, extensa documentação online e uma gama de bilbiotecas disponíveis para download, Python se torna uma ferramenta interessante para projetos complexos como uma máquina de busca. Sua principal desvantagem está no fato de ser uma linguagem interpretada e devido a este a fato possuir maior tempo de execução que linguagens compiladas como C e C++.

**3.1 Instalação e Configuração**

É necessário instalar a linguagem de programação, bem como as bibliotecas não nativas utilizadas no desenvolvimento do coletor que serão necessárias para sua execução. A seguir, os comandos usados para instalação das dependências em ambiente linux:

1. Python 3.6.0
   * *$ sudo apt-get install python3.6*
2. Gerenciador de pacotes Pip
   * *$ sudo apt-get install python-pip3*
3. matplotlib
   * *$ sudo apt-get install matplotlib*

ou

* *$ pip install matplotlib*

1. networkx
   * *$ pip install networkx*
2. html.parser
   * *$ pip install html.parser*

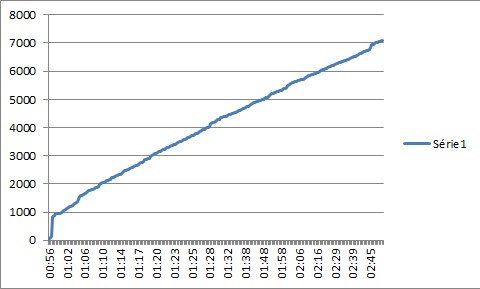
Após a instalação, os seguintes comandos podem ser utilizados para testar:

1. *$ python -V*
2. *$ pip –V*
3. *$ pip show matplotlib*
4. *$ pip show networkx*
5. *$ pip show html.parser*

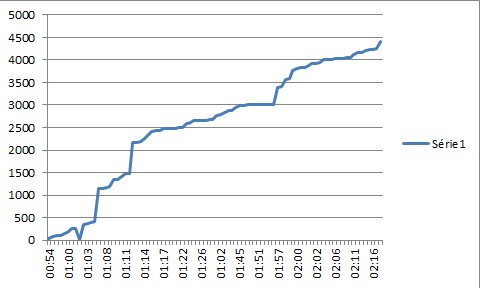
Após instalação e testes, navegue até o diretório fonte dos arquivos e execute o arquivo “engine.py”.

**4 RESULTADOS**

A taxa de coleta obtida foi de ~66 URLs por segundo. Tempo menor do que o esperado para o coletor. Tal solução proposta talvez esteja demandando tempo demais com a manipulação provisória de arquivos. Melhorias serão implementadas no armazenamento das urls através de um banco NOSQL real time chamado RethinkDB. Acredita-se que com tal mudança da taxa de coleta tenha speed up.



URLs descobertas x tempo decorrido (8 threads)

****

URLs descobertas x tempo decorrido (4 threads)