Initial Plan

Concurrent Thread-based Web Crawler

Michael Graham

MODULE NUMBER: CM0343 (40 CREDITS)

Supervisor: David W Walker
Moderator: Paul L Rosin

Project Description

In this project I will develop a multi-threaded Java application for crawling the Internet (Web Crawler). The crawler application will have the ability to process each individual web page in an appropriate manner depending on its content and structure.

The application will support a range of different configurations for constraining a crawl – for example limiting the crawl to one specific domain or specific content by providing MIME types.

Project Aims and Objectives

The objective for the Interim report due on the 14/12/2012 is to provide extensive research on the supplied areas below – as well as a minimal / partial implementation of the web crawler to further improve and extend ready for the final report.

The objective for the Final report due on the 03/05/2013 is to have a fully implemented web crawler, along with analysis of my implementation (execution times, algorithm comparisons, scheduling techniques) – as well as any future work regarding the final state of my project.

Research:

- Existing web crawlers
    - **Public-domain crawler techniques** – research relating to known crawling techniques currently in use by both proprietary and open web crawlers

- Web Crawling algorithms
    - **Path-ascending crawling** – some background information and effectiveness of ascending to every path on a given domain

    - **Focused Crawling** – some background information and effectiveness of crawling based on page similarities

    - **Page selection policies** – research regarding the best methods to state when a page should be crawled, and when it should be skipped

    - **Page re-visit policies** – research to determine when an appropriate time to re-crawl a page for changes may be. This includes determining the freshness of a crawled page, as well as the age of a crawled page.

    - **Crawl politeness policies** – research to determine the best methods to respect server administrators resources, looking specifically at a partial solution of robots.txt

- Existing architecture implementations & execution environments

Architecture:

- Parallelization
  - **Maximize page download rates** – design a efficient to maximize page crawls and downloads
  - **Minimize overheads** – ensure any parallelization implemented within the crawler doesn't cause significant overheads, e.g. for downloading a copy of a page simultaneously on separate threads
- Scheduling techniques – decide which scheduling method is most suited and if time permits, implement more than 1 scheduling technique and compare.
  - **Bredth-first scheduling**
  - **Performance based scheduling**
  - **Quality based scheduling**
  - **Crawl-ability based scheduling**
- Storage
  - **Design best storage method** – to determine the freshness and age of a crawled page, a suitable, efficient storage method is required
  - **Usage of a HashTable**– for a preliminary implementation of the web crawler, a hashtable will be sufficient to store crawled URLS. However efficiency will be scrutinized as the crawler matures
- Class diagrams - to develop an expandable, modular and efficient application within the Java environment. Also to determine object dependencies within the applications architecture.

Implementation:

- Java environment
  - **Threading and Synchronization** – make use of Java's concurrency library to achieve parallelization and efficiency
  - **New I/O** – implement and make use of Java's New I/O (also referred to as NIO) for downloading web pages.
- Specific crawler properties
  - **Different crawl method implementations** – if time constraints permit

    - **Acknowledging robots.txt** – concrete implementation of parsing web administrators crawler rules, abiding to a politeness policy

    - **Search constraints** – implementation of either customizable ".properties" file or command line arguments to control the web crawlers search constraints

Testing:

- Test-cases
    - Carry out sufficient testing on all aspects of crawler components. Provide evidence each component performs correctly.

- Execution times, efficiency
    - Document execution times throughout the project, and compare with previous implementations as well as different crawling / scheduling techniques if time permits