A better webcrawler in Go

Multimedia Information Retrieval

Neal van Veen

May 13, 2015

Advantages in Go

- Static compiled
- Rapid deployment (think Docker)
- Very simple and easy to learn
- Concurrency primitives
- Tooling

Disadvantages

- Not very flexible in syntax
- Sometimes a bit too simple
- Not as safe as needed when dealing with concurrency
- Not as fast as C/C++
- Language is still very young, needs improvements

Challenges

- Aforementioned safety is a problem
- Hashmaps used are not thread-safe, deadlocking remains an issue
- New way of thinking when programming for concurrency
- No real idiomatic way of handling errors

Data structures

- Implemented simple linked list, binary tree and fragment tree.
- Fragment-tree does not conform to Container interface, so haven't used it yet
- Using a two thousand element-big hashmap for keeping track of robots.txt
- When it is full, it is cleared, so we can somewhat buffer the contents without storing everything
- Developed a unit-testing framework for using a MongoDB backend
- No testing done for actual communication with backend over a network
- Also use a mock storage (a simple hashmap) for unittesting, that conforms to the same Storage interface.

Results

- Developed a simple Go program that generates links on a local network with a 20ms delay, so we don't pester certain websites (the LIACS websites, for instance) with too much testing HTTP requests.
- Sequential crawling of 5 minutes for 2000 requests.
- Concurrent crawling with 10 goroutines of 40 seconds for 2000 requests.
- Concurrent crawling with 100 goroutines of 6 seconds for 2000 requests.
- More goroutines testing planned, running on better hardware.
- Will be running a crawling session with a large (1000ish, hopefully) on the open net.
- Storage requirements for datastructures haven't been analyzed yet.