1. Document Search

The goal of this exercise is to create a working program to search a set of documents for the given search term or phrase (single token), and return results in order of relevance.

Relevancy is defined as number of times the exact term or phrase appears in the document.

Create three methods for searching the documents:

* Simple string matching
* Text search using regular expressions
* Preprocess the content and then search the index

Prompt the user to enter a search term and search method, execute the search, and return results. For instance:

Enter the search term: <user enters search term>

Search Method: 1) String Match 2) Regular Expression 3) Indexed

Search results:

File2.txt – X matches

File1.txt - X matches

File3.txt – X matches

Elapsed time: 40 ms

**Three files have been provided for you to read and use as sample search content.**

**Run a performance test that does 2M searches with random search terms, and measures execution time. Which approach is fastest? Why?**

See TermSearchAnalysis.xlsx.

The indexed term search was by far the fastest and also scaled much better. This is because the lookup time for each request is O(1) whereas the lookup time for the string and regex methods are O(n), as they have to parse through the entire content of the file at least once. There is a tradeoff though – keeping an indexed list does involve using more memory, which could be a consideration on more constrained machines.

**Provide some thoughts on what you would do on the software or hardware side to make this program scale to handle massive content and/or very large request volume (5000 requests/second or more).**

Handling a massive amount of requests at once could be solved in a couple of ways. The most straightforward way to me would be to launch separate threads that each kickoff term lookup work and return the results independently. This is going to be more constrained based upon the computer’s specs and in all likelihood might not be enough. The other option is to possibly split that load among multiple machines with a load balancing middle-man directing where requests go.

My implementation for an indexed search searches for all sorts of terms but if we assume search terms are whole words, then we could increase performance even more by only indexing whole words instead of letters at a time. This would improve performance by quite a bit.

In terms of hardware, my indexed method handles 2M hits in about 30000ms or about 66,666 requests per second, which would be more than enough to handle 5000 requests per second. With incorporating load balancing and possibly multiple threads, I don’t imagine the hardware requirements would be too stringent but some things I would consider:

If hardware specs was a limiting issue, could we put computers on a Hadoop cluster and leverage computing power that way? A plus to using a Hadoop cluster is that it is highly scalable because you could add or remove nodes depending on your projected usage.