Einas Madi

July 5 2016

Target Case Study

Document Search

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

After running 2M searched with random search terms, I found that the average:

```
ProcessedSearch time is faster by a factor of:
4.503 compared to the SimpleSearch
5.731 compared to the RegExSearch
```

The processed/indexed search approach is fastest. The main reason for this is that with the simple and regex search, a file is being opened and read at the same time a search for a match is being done. With a processed search, however, the file has been read, analyzed and stored in a HashMap<String,HashMap<String,LinkedHashSet<Integer>> and I'm simply checking whether certain values are present in my map. For the processed search, I've chosen to store characters in a HashMap with characters as they key and a LinkedHashSet of the locations of the key in the file as the value. This is stored as the value in a HashMap with the filenames as the key. And with a HashMap, speed is O(1) for a get, unlike the O(n) running time for SimpleSearch and RegExSearch.

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).

On the hardware side, one can use larger storage, increased RAM, take advantage of a multi-processed/multi-cored system, use caching techniques and usage of indexed databases such as NoSQL or MongoDB database.

On the software side, a user can partition the files that are being read into fixed size portions. Each portion is represented by a data structure which is serialized and stored on the hardware after pre-processing. When a user executes a search, each file (which represents, in my case, a filled HashMap<String, LinkedHashSet<Integer>>) is de-serialized, and given to the Search class with the search phrase. The data structure is then discarded after the search is complete and the next file is de-serialized. If a multi-threaded platform is used, one thread can be utilized to de-serialize files while another thread performs the search using the de-serialized data structure.