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

**Sample Result run below:**

Performance Tests: Scenarios and Results

|  |
| --- |
|  |
|  |  |
|  | /\* |
|  | Performance test that does 2 Million searches with random search terms, and measures execution time. |
|  | \*/ |
|  |  |
|  | Running Indexed test... |
|  | Time taken: **1799 ms** |
|  |  |
|  | Running String Match test... |
|  | Time taken: **17727080 ms** |
|  |  |
|  | Running Regular Expresession test... |
|  | Time taken: **11284659 ms** |

The Index search will be the fastest especially the huge number of searches done (2 million searches). As you can see from the test run results from the above. I think of this the same why for example; we put all our Jobs in SAP (RMK) in Solr; solr is our indexing software. We wanted a way for our server to get huge performance out of document searching.

Simple reason – Index Search opens a file and stores each word with frequency in that file. Next time when you search, pre-processed results are used instead of searching through the files. This is very efficient when you’re doing a large amount of search.

**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).**

**Software Improvement:**

* Code Review – to evaluate approach
* See if I can reduce “new” keyword. In Java anything creates (object) from new will cause a performance drop. I will try to use fewer ‘new’…
* Using third-party software to do indexing; for example, Solr; this will have huge performance out of document searching. Solr for example give enough configuration to tweak the processing if needed.
* Job processing/Java maint is another way I can get performance out of the document search; it allows to perform the pre-processing of search document before it is search.
* Files in database or xml format…