Application and Method of Apache Lucene

Apache Lucene is a popular open source full-text search toolkit. This article will include the introduction of Apache Lucene, basic usage and other basic content. Considering the open source nature of Apache Lucene, its performance has room for improvement, and this article will also include a summary of the latest literature on Apache Lucene's performance improvement.

1. Introduction and Application of Apache Lucene

Apache Lucene is the most popular open source full-text search toolkit, written based on JAVA language. At present, the open source search engines based on this toolkit are mature and well-known ones are Solr and Elasticsearch. After 2010, the Lucene and Solr projects were produced by the same Apache Software Foundation development team, so usually the versions we see are synchronized. The difference between the two is that Lucene is a toolkit, while Solr is an enterprise-level search application based on Lucene. In addition, our commonly used Eclipse, the search function of the help system is also implemented based on Lucene.

Apache Lucene can be used to retrieve bibliographic records, but also to search for Android bugs (M Borg, 2014), news retrieval (SS Aubakirov, 2017), research on full-text query search services for smart power distribution systems (Z Youzhuo, 2020), knowledge bases The fast peptide matching service (C Chen, 2013). It can be seen that Apache Lucene has a wide range of applications and has good research value.

2. Apache Lucene User Guide

In our daily life, the Chinese dictionary is very similar to the full-text index. Let's take the phonetic lookup method as an example. First, we find the page number of the word we want to look up through Pinyin, and then turn to the page to read the detailed explanation of the word.

In the above example, we mentioned two elements: one is the dictionary, and the other is the process of looking up words. Corresponding to the functions of Lucene, one is that we need to build a dictionary, which is called building an index, and the other is to query based on the index according to the search term.

2.1 Create an index

1) Preparation of documents (Document)

A document is the original text we want to search for.

2) Tokenizer

The document in the first step is word-cut, punctuation is removed, and useless words, such as "is", "of", etc. are removed. Commonly used open source Chinese word segmentation components include MMSEG4J, IKAnalyzer, etc. The cut words are called Tokens.

3) Linguistic Processor

The word units obtained in the previous step are processed, such as uppercase to lowercase in English, plural to singular, past participle to original form, etc. The result obtained at this time is called a word (Term)

4) Index component

The index component generates an index and a dictionary for the words obtained in the previous step, and stores them on disk. The index component first turns Term into a dictionary, then sorts the dictionary, and merges the same words after sorting to form an inverted list. Each word is stored in the list with the corresponding document Id (Document Frequency) and how many times the word appears in this document (Term Frequency).

2.2 Search

1) Enter query words

2) Lexical analysis and language processing

Split the input words, keyword recognition (AND, NOT), etc. Linguistic processing is performed on the split tokens in the same way as when building a dictionary. A syntax tree is generated from keywords and processed words.

3) Search the index to get documents that conform to the syntax tree

For example, in the syntax tree formed by A and B not C, the document list containing A, B and C will be searched, and then the document list of A and B will be used as the intersection.

4) Sort search results based on relevance

Through the algorithm of the vector space model, the correlation of the results is obtained. A relatively simple implementation is described as follows: When indexing, we get the Document Frequency and Term Frequency. The higher the Term Frequency, the higher the relevance of the document; the higher the Document Frequency, the weaker the relevance. This algorithm can be implemented by itself.

5) According to the above sorting results, return the document.

3. Improvements to Apache Lucene

Liu R(2022) proposed a high-performance MMH-index hybrid index structure is used in Apache Lucene. MMH-index can optimize the classic inverted index in Lucene: it reduces redundant data and average length in the index by adding a modal bitmap to the inverted index's dictionary. This document compares the optimized Apache Lucene with the unoptimized one, and the MMH-index significantly improves the space consumption and multi-mode query performance of Apache Lucene.

Y. Zhang (2019) proposed that Lucene's basic retrieval sorting algorithm is based on the Vector Space Model (VSM), and Lucene's Index Writer provides three parameters to adjust index performance, so adjusting these three parameters can achieve better results performance. They were adjusted according to the following formula:

$$Score = k_v{}^* VSMScore + k_p{}^* PRScore$$

The new algorithm is not entirely dependent on any one factor. It can improve the rationality by adjusting the weight coefficient, making it more flexible and reasonable than VSM.

4. References

[1] Białecki, A., Muir, R., Ingersoll, G., & Imagination, L. (2012, August). Apache lucene 4. In *SIGIR 2012 workshop on open source information retrieval* (p. 17).

[2] Borg, M., Runeson, P., Johansson, J., & Mäntylä, M. V. (2014, September). A replicated study on duplicate detection: Using Apache Lucene to search among Android defects. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical*

Software Engineering and Measurement (pp. 1-4).
[3] Liu, R., Liang, J., Jin, P., & Wang, Y. (2022, October). MMH-index: Enhancing Apache Lucene with High-Performance Multi-Modal Indexing and Searching. In *Proceedings of the 30th ACM International Conference on Multimedia* (pp. 7279-7289).
[4] Zhang, Y., & Li, J. L. (2009, August). Research and improvement of search engine based on Lucene. In *2009 International Conference on Intelligent Human-Machine Systems and Cybernetics* (Vol. 2, pp. 270-273). IEEE.
[5] Hirsch, L. (2010, July). Evolved Apache Lucene SpanFirst queries are good text classifiers. In *IEEE Congress on Evolutionary Computation* (pp. 1-8). IEEE.
[6] Ji, W. (2020, June). Research and Application of Information Data Retrieval System in Station Based on Lucene Technology. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)* (pp. 687-690). IEEE.
[7] Azzopardi, L., Moshfeghi, Y., Halvey, M., Alkhawaldeh, R. S., Balog, K., Di Buccio, E., ... & Palchowdhury, S. (2017, February). Lucene4IR: Developing information retrieval evaluation resources using Lucene. In *ACM SIGIR Forum* (Vol. 50, No. 2, pp. 58-75). New York, NY, USA: ACM.
[8] Chen, C., Li, Z., Huang, H., Suzek, B. E., Wu, C. H., & UniProt Consortium. (2013). A fast peptide match service for UniProt knowledgebase. *Bioinformatics*, *29*(21), 2808-2809.
[9] Lucene, A. (2010). Apache lucene-overview. *Internet: http://lucene. apache. org/iava/docs/[Jan. 15, 2009]*.
[10] Milosavljević, B., Boberić, D., & Surla, D. (2010). Retrieval of bibliographic records using Apache Lucene. *The Electronic Library*.
[11] Carlson, P. (2006). Apache Lucene-Query Parser Syntax. *Obtenido de< https://lucene. apache. org/core/2_9_4/queryparsersyntax. pdf>(21 de 06 de 2013)*.