International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 12 (2015) pp. 32103-32112
© Research India Publications
http://www.ripublication.com

# Efficient Content Sorage And Just In Time Retrieval In Cloud

**S.Sageengrana[#1], P.Ponnaruvi [#2], G.Tamilmani [#3] and K. Rajathi**

*Department of Computer Science and Engineering*
*[#1, #2, #3] Assistant professor*
*[#1, #2]Vel Tech High Tech Dr.RangarajanDr.Sakunthala Engineering College*
*[#3]Vel Tech Technical University*
*No.60 Avadi-VelTech Road, Chennai-600 062, TamilNadu, India*
granadhas@gmail.com,, taman.maya@gmail.com, goldfallz@gmail.com
k.rajathimtech@gmail.com

**Abstract**

The size of the databases used in today's enterprises has been growing at exponential rates day by day. Simultaneously, the need to process and analyze the large volumes of data for business decision making has also increased. Storing a lot of content in the cloud extends waiting time and increases search time. Distributed file systems (Hadoop) are key building blocks for cloud computing applications based on the Map Reduce programming paradigm. It efficiently solves this problem. In this paper proposes an efficient content storage and retrieval based on keyword ranking and BIDD(Bitmap Index Database Design) technique in distributed environment. Proposed keyword ranking first extract nouns than give ranks for each extracted noun. BIDD techniques based on bitmap representations are known to improve query response time in a large environment. Hbase (database) of Hadoop is used to store the large content. This paper reports the experimental work on large content storage and its optimal solution using Hadoop database, Hadoop Distributed File System (HDFS) for storage and using parallel processing to process large data sets using Map Reduce programming framework.

## I. INTRODUCTION

In this electronic age, increasing number of organizations are facing the problem of explosion of data and the size of the databases used in today's enterprises has been growing at exponential rates. Data is generated through many sources like business processes, transactions, social networking sites, web servers, etc. and remains in structured as well as unstructured form [2]. It is not possible for single or few

machines to store or process this huge amount of data in a finite time period. Processing or analyzing the huge amount of data or extracting meaningful information is a challenging task.

## II RELATED WORK

Large amount of content stored in distributed environment. If there is an increase in the level of content it's also increase the delay of content retrieval. If content is stored properly only then it is possible to retrieve content just in time. The research work is reviewed in two subsections. In the first subsection, we provide a review of related work on the keyphrase extraction from large content. Bitmap indices in second subsection which is the focus of parallel processing.

### A. KEYWORD EXTRACTION

Keyphrases extraction and Bit map table creation are the two important methodologies for efficient content storage and retrieval. [6]Keywords provide a concise and precise high-level summarization of a document. An important feature for document retrieval, classification, topic search and other tasks even if full text search is available. Introduced kea algorithm for automatic keyphrase extraction based on keyphrase assignment and keyphrase extraction[8]. Calculating the correlation between words of each document, to determine the keywords that give out the fields of interest of each document content[3]. An information theoretic explanation of tf.idf is given by [9]. In [10] 4 different features are used for keyword extraction: term frequency, collection frequency, relative position of the (first occurrence of) the word in the text, and number of times a term is used as keyword. In this paper we propose a keyphrases extraction based on noun base noun is reflect whole meaning of particular sentence so its provide suitable keywords for a particular document.

### B. INDEXING STRUCTURE

Indexing is the optimal method for improving the speed of returns on queries without adding additional hardware**.** In[1] μ-Tree is a higher efficiency index. The μ-Tree is improvement of B+-tree for flash memory. In recent years, numerous tree-based indexes (e.g., distributed Btree index [7], [4] and distributed R-tree index [8], [5]) have been proposed for Cloud systems. These indexing schemes offer good query performance, but are not space efficient and costly to maintain. This is because 1) the index size is usually proportional to, sometimes even larger than the data size itself; 2) to support various types of queries, a large number of indexes have to be built on different attributes, incurring more overhead. Due to the fact that data volume is extremely large in the Cloud, building indexes for these data yield unacceptable cost, as a large number of compute nodes have to be purchased to maintain the indexes. Therefore, to provide scalable data retrieval service in the Cloud, we need to re-examine how indexes should be designed. In this paper, we propose BIDD (Bitmap Index for Database Design), a specialized bitmap indexe for a large-scale data store. BIDD is built on top of the underlying DFS(Distributed File System), and adopts a set of techniques to make bitmap indexes more scalable. Only one MapReduce job is

required to build the indexes for the columns of one table. Compared to tree-based indexes, BIDD can be built/rebuilt very efficiently.

## III PROPOSED WORK

The proposed work is solves the problem of large content storage and retrieval in cloud. Fig 1: illustrate how keyword methodology and BIDD is embedded into existing data storage system. Our goal is to develop efficient system for store large content.

The BIDD index contains three modules (update handler, index tracker process, index handler). BIDD interacts with external system like HadoopDB(Hbase). The index tracker processor is build and maintain the BIDD. When data are imported into the system, the index manager create the index. To handle infrequent update by update handler. When a query is enter the query handler process the query.

The keyword extraction four modules (content preprocessing, morphological analyzer, NP extraction, ranked keyword). The content pre processing stage remove all numerical, stop words, stemming etc.. Morphological analyzer split the sentence like verb, noun, adverb etc.. NP extraction only extracts nouns from splited sentence. Last stage of keyword extraction is Ranking the keyword based on TF*IDF value.

Base on the content similarity it's creating distributed learning object. Both the BIDD and the imported data stored in the DFS.

## NOUN PHRASE EXTRACTION

The first and foremost step in efficient content storage is keyword extraction from document collection.

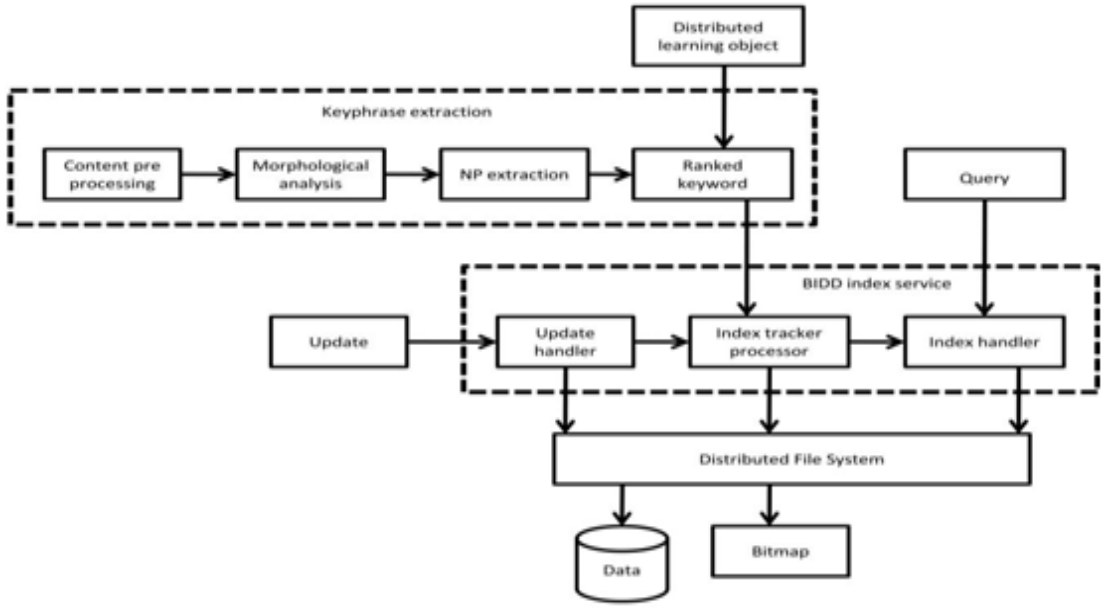


**Fig 1: System design**

Keywords provide a concise and precise high-level summarization of a document. They therefore constitute an important feature for document retrieval, classification, topic search and other tasks even if full text search is available. Fig 2 shows various steps for extract keywords from text document. The document is first converted into a text document if document is not in.txt format. In the pre processing stage, removal of the symbols, numerical values and stemming take place. Morphological analysis split each sentence based on verb, noun, adverb etc.. Finally only nouns are extracted from divided sentence and are ranked. Because nouns have a all information about particular sentence.

Morphological analysis is a Part-Of-Speech Tagger (POS Tagger) that read a text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc..

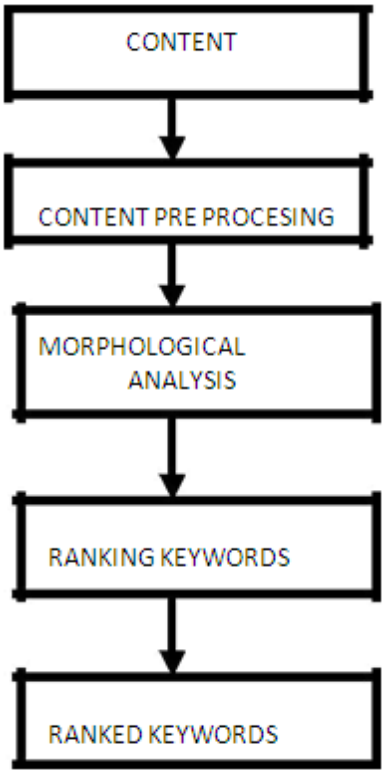


**Fig.2 Keyword Extraction**

This tagger output is useful to extract only nouns from collection of words. Keyword is arranged based on the priority value.It is helpful during the search time avoid to search entire table. Based on Term Frequency and Inverse Term Frequency the extracted nouns are prioritized. It compares the frequency of keyphrase's used in a particular document and collection of document.

The TFxIDF for phrase P in document D is,

$$\text{TF}\times\text{IDF} = \frac{\text{freq}(P, D)}{\text{size}(D)} \times -\log_2 \frac{\text{df}(P)}{N}$$

The freq(P, D) is the number of times P occurs in D, size(D) is the number of words in D, df(p) is the number of documents containing Pin the global corpus and N is the size of the global corpus. All extracted keywords are arranged in descending order of value. D is a document, D.W is a particular word in a document

**Algorithm 1**(document D, word W)

1. **if** D is a document **then**
2. build the morphological analyzer

3. **else**
4. **if** f(D.W) is noun **then**
5. Rank the keyword(strategy 2)

6. **else**

7. **if** f(D.W) is verb **then**
8. Remove the verb(strategy 3)

9. **else**
10. **if** (D.W) is numeric **then**
11. Remove the numeric(strategy 4)

12. **else**
13. No keyword is extracted

**BIDD OVERVIEW**

A Bitmap index is a special kind of index that stores the bulk of its data as bit array (bitmaps) and answers most queries by performing bitwise logical operations on these bitmaps..

**a) Dynamic index formation**

The pattern based index formation helpful to reduce the search time. Patterns are having two keyword group. Each keyword group maintains particular keyword related all the materials.

The pattern formation based on keyphrase priority. This index table is creating dynamically based on content storage. The table 1.1(a) explains about pattern based

dynamic index formation, Attribute A is set of the bitmap vector p{p0, p1, p2, …….pn-1}. k is the number of keywords extracted(k0, k1, k2, ….kn).A bit at position i in the bitmap vector is set to 1 if the record at position i in the indexed table is satisfied

$$n = \sqrt{2} + 0.25 + 0.5$$

$$D^{j} = \begin{cases} 1 \text{ if and s} \\ 0 \quad \text{otherwise} \end{cases}$$

Mapping table formed dynamically based on result of pattern formation. It is helpful to identify the correct keyword group from pattern. The result of the pattern based index table mapped to mapping table find out the correct keyword group. Fig 3 explain the overview structure of mapping table and pattern table. Single bitmap index table maintain keyword group and point to the Meta data file location.

Consider user keyword is stack the following algorithm process this keyword then give a meta data files

| Value | Keyword | P0 | P1 | P2 | P3 | P4 |
|---|---|---|---|---|---|---|
| 1<br>0 | Array<br>Algorithm | 0 | 1 | 1 | 0 | 0 |
| 1<br>0 | Binary<br>Expression | 0 | 0 | 0 | 1 | 1 |
| 1<br>0 | Implement<br>Input | 0 | 1 | 0 | 1 | 0 |
| 1<br>0 | Queue<br>Stack | 1 | 0 | 0 | 0 | 1 |
| 1<br>0 | Static<br>Tree | 1 | 0 | 1 | 0 | 0 |

| Map | Keyword |
|---|---|
| 0 | Array<br>Algorithm |
| 1 | Binary<br>Expression |
| 1 | Implement<br>Input |
| 0 | Stack<br>Queue |
| 1 | Static<br>Tree |

**Fig 3: pattern and mapping table**

| Keyword | I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Array | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Algorithm | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Binary | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Expression | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Implement | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Input | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Queue | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Stack | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Static | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Tree | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Fig 4: Group finder table**

## VI. EXPREMENT AND RESULT

The fist experiment was text preprocessing, count the number of words occur within a set of large sized document and extract the ranked keywords. The automatic keywords extraction involves reducing a text document or a larger corpus of multiple documents into a read whole document than find out important information. Fig 5: result compare auto keyword with manually selected keyword.

| Text | Automatically selected | Manually selected | Matches | precision |
|---|---|---|---|---|
| 1 | 4 | 3 | 3 | 75% |
| 2 | 9 | 8 | 6 | 63% |
| 3 | 8 | 8 | 6 | 71% |
| 4 | 10 | 10 | 8 | 82% |
| 5 | 10 | 10 | 9 | 90% |
| 6 | 5 | 4 | 4 | 64% |
| 7 | 7 | 5 | 3 | 55% |
| Over all precision | | | | 71% |

**Fig 5: Automatic keyword extraction**

The second experiment was BIDD storage in Hadoop(Hbase) database and Mysql database. The evaluation fig 6: result shows Store less amount of content in database the mysql is give good performance than Hbase but storage of content is increase the Hbase give good performance.

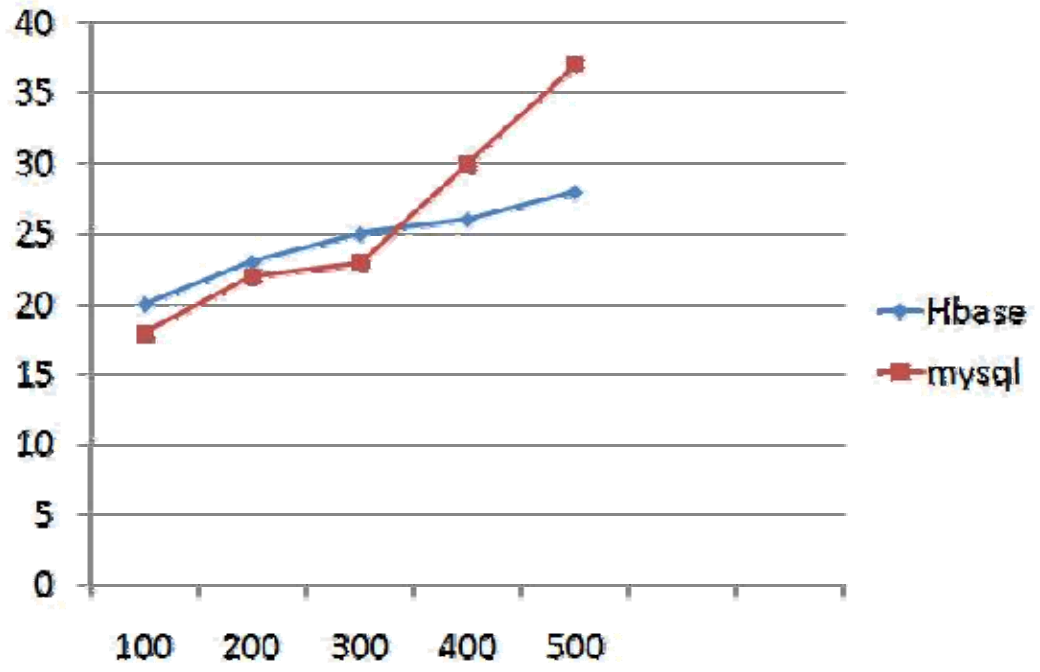


**Fig 6: query performance and scalability**

## V. CONCLUSION

Our BIDS index is storage efficient and easy to maintain, which makes it more scalable. In this paper, a keyword extraction, bitmap based indexing scheme, BIDD, Hadoop database is proposed to manage large amount of data in the Cloud. Our keyword extraction BIDD index is storage efficient and easy to maintain, which makes it more scalable. In the future, the query operators are transformed into a set of bit-wise AND/OR operators, which can be handled more efficiently and Metadata management in HDFS and load balance in hadoop data nodes is to be concentrated.

## REFERENCE

[1] Junhua Fang, Hanhu Wang, Mei Chen Dan Ma, "A Self-adaptive Improved µ-Tree Index Structure for Flash-based DBMS" International Conference on Systems and Informatics (ICSAI) **2012**

[2] Impetus white paper, March, 2011, "Planning Hadoop/NoSQL Projects for 2011" by Technologies, Available:http://www.techrepublic.com/ whitepapers/planninghadoopnosql-projects-for-2011/2923717, March, 2011.

[3] Saber Heni, Ridha Ejbali "A Neural Principal Component Analysis for text based documents keywords extraction" 3rd International Conference on Next Generation Networks and Services, **2011**

[4] S. Wu, D. Jiang, B. C. Ooi, and K.-L. Wu, “Efficient b-tree based indexing for cloud data processing, ” *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 1207–1218, 2010.

[5] J. Wang, S. Wu, H. Gao, J. Li, and B. C. Ooi, “Indexing multidimensional data in a cloud system, ” in *SIGMOD*, 2010, pp. 591–602.

[6] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin and Craig G. Nevill-Manning, “KEA: Practical Automatic Keyphrase Extraction”, International Conference on Communication and Information Technology, **2008**

[7] M. K. Aguilera, W. Golab, and M. A. Shah, “A practical scalable distributed b-tree, ” *VLDB*, 2008.

[8] C. Mouza and W. Litwin, “Sd-rtree: A scalable distributed rtree, ” in *ICDE*, 2007, pp. 296–305.

[9] A. N. Aizawa, “An information-theoretic perspective of tf-idf measures, ” Inf. Process. Manage., vol. 39, no. 1, pp. 45–65, 2003.

[10] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, “Domain-specific keyphrase extraction, ” in IJCAI, T. Dean, Ed. Morgan Kaufmann, 1999, pp. 668–673.