Efficient Document Placement in Hadoop

Pratik Thakkar, Saurabh Thakre, Prakash Thete, Ajinkya Wani, D.D. Gatade

Department of Computer engineering,
Sinhgad College of Engineering,
Pune -41, India

pratikthakkar11@gmail.com, thakresaurabh009@gmail.com, prakashthete777@gmail.com, ajinkyawani@gmail.com, ddgatade.scoe@sinhgad.edu

Abstract: **Hadoop is an open source java based implementation of Google's map-reduce framework. It is implemented for applications which are highly parallelized and use large clusters. Map-Reduce is a programming model which is used for processing large data sets.Programs written in this style are automatically parallelized and can be run on large clusters of commodity machines. Parallel computation makes the processing faster and reduce the time complexity. Hadoop framework is known for its high fault tolerance and capacity to work upon terabytes of data.**

We will process the large data for document similarity. It is possible to form desired document clusters according to similarities in their content. This will help to reduce manual overhead of sorting the data and efficient storage and retrieval of information.

Keywords-Hadoop, Map-Reduce, Cluster, Keyword Identification, Cluster Formation

I. INTRODUCTION

Information explosion is the rapid increase in the amount of published information and the effects of this abundance of data. As the amount of available data grows, the problem of managing the information becomes more difficult, which can lead to information overload. The World Wide Web saw a revolution with the advent of Web 2.0. Web applications became more interactive, allowing users the freedom to interact and collaborate over the Internet in ways not possible earlier. Users started to take the role of content creators rather than passive viewer of web pages. Websites started to get swamped with user-generated content from blogs, videos, social media sites and various other Web 2.0 technologies. This had the direct consequence of information stored on servers exploding into sizes not seen earlier. Contributing to this information explosion was the already accumulating business data that was generated everyday across various companies and industries. These were just a couple of reasons that led to the age of Petabytes - an age where information stored in data stores reached levels of Petabytes or 1024 Terabytes or 10^6 Gigabytes. With more and more enterprises using computers and embracing the digital age, more and more information is starting to get digitized and the volume of data published and stored is increasing with each passing day.

Thus we are offering a system for easy retrieval, manipulation and storage of data. We have used multiple commodity machines for the processing of data.

II. LITERATURE SURVEY

Map-Reduce style of computation is used for processing huge amount of data and is highly parallelized. In [1], authors have addressed the need of a system to manage the exponentially growing data. Hadoop provides a framework to process large amount of data in parallel style. In [2], authors have focused on basic usability of map and reduce function. It explains how the key value pair is generated and term frequency is generated. [3] mainly discusses on Hadoop framework and the HDFS.HDFS is very large file system which uses commodity machines and is highly fault tolerant.[4] explains the evaluation of running the map-reduce algorithm. Removing the stop-words from the input file increases the efficiency of the algorithm as the number of tuples emitted from the mapper are reduced.

III. SYSTEM DESIGN

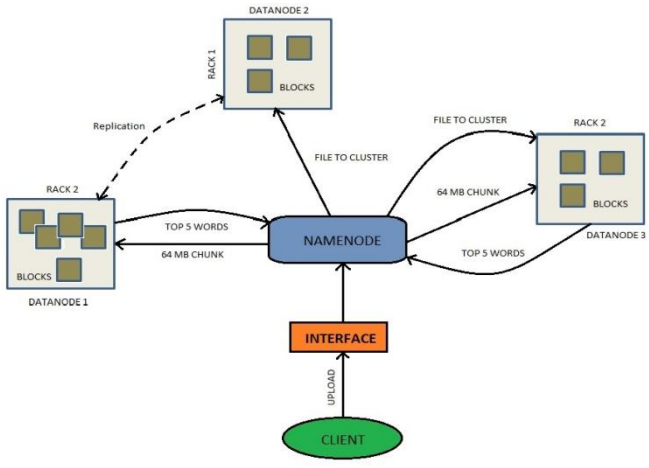

Fig. 1. The architecture overall system

A. Map-Reduce Functions

In our system, we have used four commodity machines. Client uploads the input file to be stored in the cluster. The Map-Reduce functions derive the keyword frequency of terms in the file.

The large amount of input data is divided into n no. of splits of 64 MB blocks. The copies of data are stored in racks.

There is one master (Namenode) in the cluster which assigns the tasks to the idle workers(machines other than the master i.e. Datanode)

The workers (Datanodes) parse the key-value pair and assigned it to the user defined map function also store in the memory.

The locations of the key-value pair in the memory is passed back to the master which assigns these locations to the reduce workers.

The reduce workers read the buffered data from the local memory of map workers and sort intermediate key values to group together the same key values.

Reduce workers pass the keys and its intermediate value pairs to the reduce function.

When the map and reduce functions are completed, the master wakes up user program.

B. Cluster Formation

The user program reads the top five keywords from the output file which contains the word-count of the input file.

Each cluster is recognized by its signature keyword set. The top keywords of a file are matched with the keyword set of each Cluster. If the keywords are matched with a certain cluster, then that file is stored in that cluster.

If the top keywords of the file do not match with any keyword set, then a new cluster is formed with signature keyword set as the top keywords of the input file.

IV. FUNCTIONAL UNITS

A. User Input:

The user uploads the input data which is to be stored in a cluster.

B. Input Data Block Distribution

If large size of input is uploaded, then it is split into blocks of 64 MB. Then these blocks are distributed by the master for Map-Reduce job.

C. Stop-Word Cleaning:

Words which are not related to particular files (i.e. stop-word) e.g. am, is, was, there, here etc. are removed for keyword identification phase.

D. Keyword Identification:

Maximum time occurring words except stock-word are considered as keywords of a particular file. These are useful for an identifying the type of file.

E. Cluster Identification:

Keywords of a current file is compared with a keyword of an existing cluster and depending on this cluster for file storage is decided. Maximum keyword matched cluster is selected for a storage.

F. Effective Storage:

The data files are stored depending on the content of the file. Files with the same contents are placed under one cluster on commodity machine, so in case of retrieval it's much easier to retrieve same data from same source rather than from different sources.

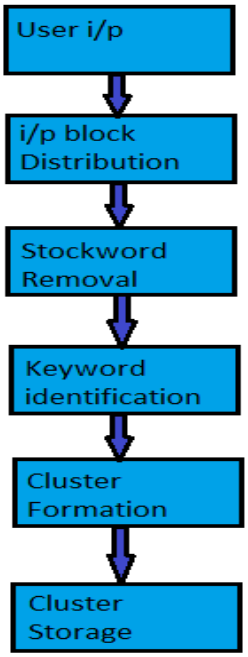

Fig. 2. The functional blocks of the system

V. MATHEMATICAL MODEL

1) Let the I/p of the system be the files to be uploaded, Let I= {f1, f2, f3...}

2) The files are stored in the node cluster, C= {C1, C2, C3…}

3) The function for clusters and their signature keyword set is, $Ck_{ey(i)=}K_1, K_2, K_{3,..}$

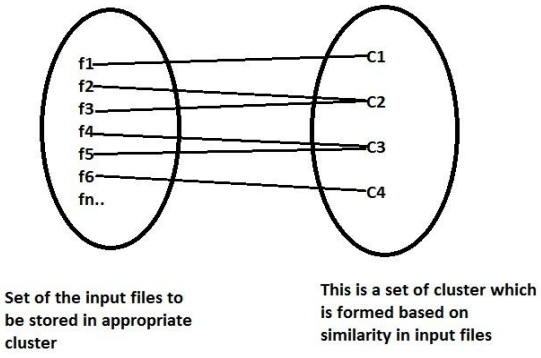

Fig. 3. Mapping of files to cluster

Explanation: Based on the Corresponding match of keyword to its cluster the particular file is stored to that particular cluster

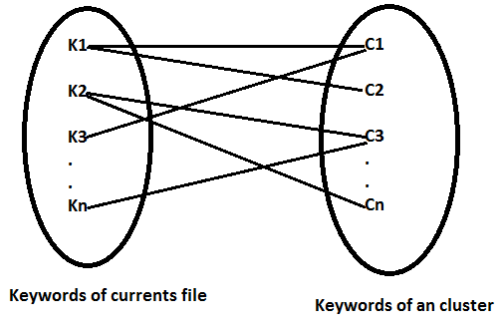

Fig. 4. Mapping keywords of files to Cluster

Explanation: This Mapping facilitates keyword match corresponding to its true identity of cluster.

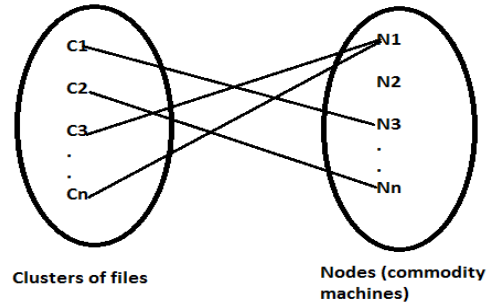

Fig. 5. Cluster storage on different machines

Explanation: This mapping shows storage of clusters on the nodes.

4) Function to map files to the cluster,

$m(f_i)=C_i$

This function distributes 64MB data chunks to the data nodes free for load bearing

5) Function for clean files with removed stock words, M_c (f_i) $=f_i'$ (Where f_i' is a cleaned file, obtained by removing stock words like a, an, the, is etc.) This function cleans files for stop-word comparing it with standard stock_base(database) .

6) Now we find key value pair from the map function, $M_{wc}(f_i')$

(Where V is a vector and V= {($word_1$, count1), ($word_2$, $count2_{),(}$ $word_3$,$count_{3)}$})

It finds the frequency of the words from the input file.

7) We have to find the most commonly occurring word from the above function, $M_{top}(V)=K$,

K is high frequency keyword,

K= {$keyword_1$, $keyword_2$, $keyword_3$, $keyword_4$, $keyword_{5,\ldots}$}, Let count of keywords be,

$keyword_1$->count1,

$keyword_2$-> count2,

If (count1>count2)

Then Max $(count_i)$ $=top_5$ (count1, word1)

Selecting top five maximum frequency keywords from the vector as key identifier's for the file.

8) Let Ck_i is our key under consideration, Comparing this key with our set of Keywords,

If($M_{compare}(K,Ck_i)$=NULL)

$F_{create}(C)=CUC_{n+1;}$

$C_{key(na1)}=Ck_i$ Else

O= {Ci}

(Putting file in the cluster Ci)

Comparing keywords with existing cluster keywords, for true place the file into identified cluster and for no match new cluster will be created and file is stored into it.

The files frequency keywords would be set as clusters identified keywords.

VI. RESULTS AND CONCLUSION

We present an algorithm based on Map-Reduce framework which will create document cluster based on content similarity of the documents. The documents which are based on related topic or same topic will be together in one cluster. Thus the final output will be consisting of multiple clusters each with its set of keywords. We are offering certain benefits like easy data retrieval and its effective storage.

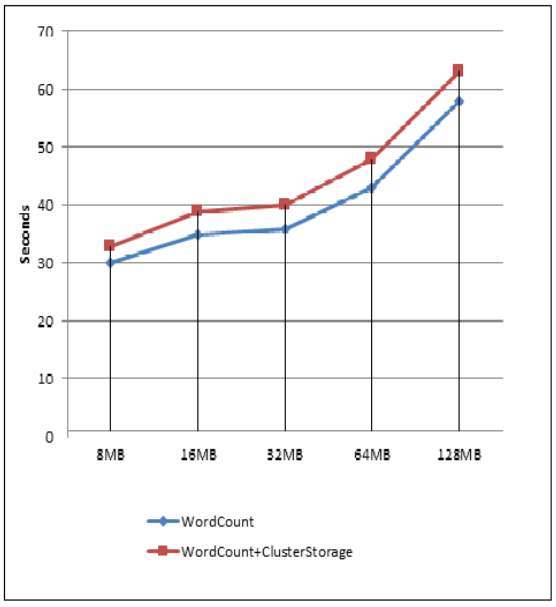

Fig. 6. It shows the time required for file storage for varying file size.

The above graph shows time required for the operation for variable input data size. This graph shows result when our algorithm is running on a single node. When the data size is more than 64 MB, it gets divided in blocks of 64 MB, and for this splitting some time is required. Thus, the graph shows little elevation after 64 MB data size for both the lines. The blue line indicates the time required for standard WordCount program. WordCount is the Map-Reduce job which finds the keyword frequency. The red line shows the graph for our system. Cluster Identification and Cluster storage takes little quantity of time for document storage. So, with some more amount of time (after the keyword frequency calculation is done), the document can be placed in a proper cluster. Thus effective storage and retrieval can be achieved.

REFERENCES

[1] Aditya B. Patel, Manashvi Birla, Ushma Nair, (2012). Addressing Big Data Problem Using Hadoop and Map-Reduce.

[2] Jeffrey Dean and Sanjay Ghemawat. Map reduce,"Simplified Data Processing on Large Clusters"

[3] Kala Karun A., Chitharanjan K., (2013). A Review On Hadoop –HDFS Infrastructure Extension.

[4] Tamer Elsayed, Jimmy Lin, and Douglas W. Oard.,(2008). Pairwise Document Similarity in Large Collections with Map-Reduce.