ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Filtered Indexing: An Alternate Indexing Mechanism for De-Duplication

S.Preethi
Computer applications
R.M.K. Engineering college
Chennai, India
spreethi.thesis@gmail.com

L.Mary Immaculate sheela*
Computer applications
R.M.D. Engineering College
Chennai, India
drsheela09@gmail.com

***Abstract*:** Years ago internet was merely used to retrieve information. Later on paradigm shifted and internet came a long way of providing different types of services to the users. Today cloud computing has become the buzz word. Cloud describes a new supplement, consumption and a delivery model for internet based services. Cloud further offers virtualized services over the internet. When the data is spread wide across the cloud, duplication becomes inevitable. It is virtually impossible to eliminate duplication. At present, there is a vast amount of duplicated data or redundant data in storage systems. Data de-duplication can eliminate multiple copies of the same file and duplicated segments or chunks of data within those files. Instead we can avoid redundancy through data de-duplication. Data de-duplication is a technique where in the redundant data is deleted keeping only the unique copy of the data. Current issue for data de duplication is to avoid full-chunk indexing to identify the incoming data is new, which is time consuming process.Thereby improving storage utilization. In current scenario Full chunk indexing is a major issue over the cloud. In this paper we propose an efficient indexing mechanism using the filtered index databases. In this paper first we divide the variable length chunks using the sliding window. Then each chunk is given a chunk ID using a hash function. The disk storage is much less than that required by a table and search time is much reduced with the use of filtered index databases.

Keywords*:* Cloud computing, Data De-duplication, Full Chunk Indexing, Filtered index.

I. INTRODUCTION

Technology has revolutionized today's world. With the advancement in technology companies these days are downsizing because computers today does a job which many men had been tiring for long hours. Cloud is a similar scenario of different kind. On-demand self-service Internet infrastructure where you pay-as-you-go and use only what you need, all managed by a browser, application or API. Cloud computing is broken up into multiple segments including: Cloud Infrastructure, Cloud Platforms and Cloud Applications [1].When everything is shared over a cloud network, then there is a tendency of data being replicated knowingly or unknowingly over the network. De-duplication is an intelligent compression technique where we try to identify duplicate entries at different location thereby reducing need for storage space [2].There are many existing systems which resolve to solve data De-duplication. But in most cases they don't take into consideration the problem of full chunk indexing. These problems can be avoided by choosing a better searching technique. In our proposed system we use the filtered indexing.

Filtered index is a technique where we index a portion of rows in a table that means it applies filter on INDEX which improves query performance, reduce index maintenance costs, and reduce index storage costs compared with full-table indexes [3]. A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of slower writes and increased storage space. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records. The disk space required to store the index is typically less than that required by the table (since indices usually contain only the key-fields according to which the table is to be arranged, and exclude all the other details in the table), yielding the possibility to store indices in memory for a table whose data is too large to store in memory [4].

The proposed algorithm is divided into following steps: Divide the incoming data stream into fixed length chunks using a time division multiplexing. Then generate chunk IDs by giving timestamps to each chunks. Here too the timestamp is fixed based on first come first served basis. Then check the chunks using filtered index. Store only the unique chunks on the storage there by eliminating redundancy.The rest of the paper is organized in the following manner. Section 2 contains the literature works. Section 3 contains the motivation for the work. Section 4 contains the algorithm of the proposed system section5 contains the future works and the conclusion.

II. PROBLEM STUDIED

M.Lillibridge et al. described problem statement as chunk lookup disk bottleneck/full chunks indexing encountered in inline deduplication and they try to solve using sampling and sparse index and chunk locality [5]. However, they based on assumption that if two segment share one chunk, it is likely to be shared other chunks only limited number of segments are deduplicated and can't do fully deduplication as sometimes can store duplicate chunks. Walter Santos solved the problem of identification of replicas in database with parallel deduplication algorithm using filter-stream model [6] and only considered for databases and not for sub-file level of the file which have their respective secure hash code.

Daniel P.Lopresti described the issues related to the detection of duplicates in document image databases and

proposed four algorithm using edit distance [7]. But they can solve only for image database specific.Tin Thein Thwel, Ni LarThein in their International conference paper they had proposed an indexing technique using B^+trees[8].But the problem with the B^+ trees or in general trees tends to occupy space for empty leaf nodes.

III. MOTIVATION

Cloud computing reduces the costs to design, build, deploy, and support the products and services and share them over a wider spectrum. One cannot avoid the problem of redundancy when there is a huge volume of data being shared over a network.

There are many systems to avoid this problem of duplications and to decrease the need for storage space. But the major problem in this concept is full chunking. Moreover many of them use Hashing techniques which are CPU computative. To eliminate these issues this paper proposes a new methodology of time stamping the chunks. The incoming data stream is divided into fixed length chunks using the time division multiplexing. Then using the filtered indexing chunk IDs are checked for the uniqueness. Only the unique chunks are then stored on to the metadata. The metadata is stored in a table format thereby reducing the space required for storage. The use of filtered index accelerates the search time.

IV. DEDUPLICATION ARCHITECTURE

The proposed architecture for the de-duplication is as shown in the figure 1.

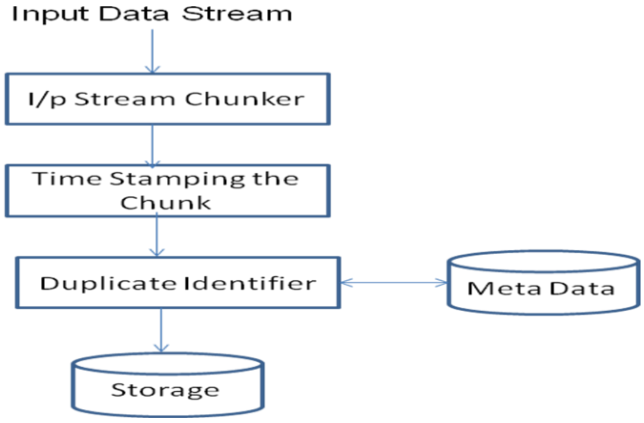

Figure:1 Architecture Diagram

A. System Component Description:

The input data stream can be in any form. It can be either a text file or a pdf file or .doc file etc.., The Input data Stream is segmented into fixed size chunks using the time division multiplexing. Each chunk is given an ID using the Timestamp. Then the duplicate identifier checks for the redundant data by checking it against the metadata. The proposed filtered index is built on the metadata. Once the chunks are checked for uniqueness then they are stored on to the storage media.The components of the proposed architecture are:

a. ***Input Stream Chunker***: The Input data stream is divided into number of fixed length chunks using the time division multiplexing.

b. ***Time Stamping the Chunks:*** The divided chunks are given unique identification. This is done by time stamping each chunk. The chunks are time stamped based on first come first served basis.

c. ***Duplicate Identifier:*** The role of the duplicate identifier is to check for the Chunk IDs with that of the meta data. Meta data: Meta data is stored in the form of the Filtered indexes. The Filtered index is stored in a table format with two columns. One the Index and other is the ChunkID which based on time stamping.

Moreover many of them use Hashing techniques which are CPU computative. To eliminate these issues this paper proposes a new methodology of time stamping the chunks. The incoming data stream is divided into fixed length chunks using the time division multiplexing. Then using the filtered indexing chunk IDs are checked for the uniqueness. Only the unique chunks are then stored on to the metadata. The metadata is stored in a table format thereby reducing the pace.

B. Input Stream Chunker:

The input data stream is divided into number of fixed length using the time division multiplexing. The Time-division multiplexing (TDM) is a type of digital or (rarely) analog multiplexing in which two or more signals or bit streams are transferred apparently simultaneously as sub-channels in one communication channel, but are physically taking turns on the channel. The time domain is divided into several recurrent timeslots of fixed length, one for each sub-channel [10].The Proposed Mechanism using filtered indexing: A filtered index is an optimized nonclustered index, especially suited to cover queries that select from a well-defined subset of data.It uses a filter predicate to index a portion of rows in the table. A well-designed filtered index can improve Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation. query performance, reduce index maintenance costs, and reduce index storage costs compared with full-table indexes [9]. Time Division Multiplexing is the process of dividing up one communication time slot into smaller time slots.

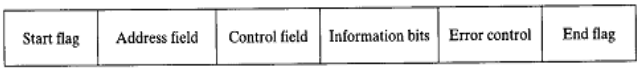

Figure 2 Statistical TDM Packet

V. CONCLUSION

Here we had tried to design a system that is more efficient in terms of space and search time. Most of the existing system uses hashing functions to generate the chunkIDs which take up a lot of CPU's time. In contrary the use of time stamps to generate chunkIDs reduces the CPU computational intensiveness. More over we are proposing a system which uses the filtered indexing. The use of indexing can decrease the search time there by enabling faster data retrieval. More over the meta data is build in a table format in contrast to B^+

trees which occupy space for empty sub trees. Use of filtering in the database may enable the user to query the metadata at the required point of time.The implementation of the proposed system is out future work. Once the implementation is accomplished it will be tested on different sets of data.

VI. REFERENCES

[1] www.servepath.com/support/definitions.php.www.information-management.com/glossary/d.html*).*

[2] http://dotnetkicks.com/database/Introduction_to_Filtered_Index_Improve_performanceI

[3] http://en.wikipedia.org/wiki/Index_%28database%29.

[4] Michael T. Goodrich, Data Structures and Algorithm in C++, Wiley Publishing, 2009, pp. 598.

[5] Sridhar Ramaswamy, "Efficient Indexing for Constraint and Temporal Databases", Preceedings of 6th International Conference on Database Theory, Springer Berlin/Heidelberg Bell, Delphi, Greece, 1997, pp. 419431.

[6] Daniel P. Lopresti, "Models and Algorithms for Duplicate Document Detection", Preceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR'99), IEEE Computer Society, Washington, DC, USA, 1999, pp. 297-300.

[7] Tin Thein Thwel, Ni LarThein , "An Efficient Indexing Mechanism for Data Deduplication" , Preceedings of Internation IE conference IEEE,2009.

[8] http://technet.microsoft.com/en-us/library/cc280372.aspx

[9] http://en.wikipedia.org/wiki/Time-division_multiplexing