Investigating Techniques for Searching Files in Distributed Systems: An Overview

[1]Vaishali S. Ambadekar, [2]Sunil R. Gupta
[1]Department of Computer Science and Engineering,
[2]Department of Computer Science and Engineering, PRMIT&R, Badnera.

Abstract*—High Performance applications generates large amount of data. The large amount of data generated from computation leads to data been dispersed over the file system. Problems begin to exist when users need to locate these files for later use. For small amount of files this might not be an issue but as the number of files begins to grow as well as the increase in size of these files, it becomes difficult locating these files on the file system using ordinary methods. This paper focuses on study and investigation of searching techniques for files in Distributed Systems. Also analyses various distributed environments and existing searching approaches for distributed systems.

***Keywords*—** *Distributed systems, searching, distributed file systems, structured-per-to-peer network, hash table, binary-search-tree.*

I. INTRODUCTION

In Modern era, computations require powerful hardware. One way of gaining results faster is getting new hardware over and over again. Buying a supercomputer is, however, not a cheap solution and takes plenty of time to install software to these new supercomputers. Also there is a lot of overhead in maintaining them. Another way in achieving better system performance is using a distributed system. In a distributed system, several computers are connected together usually by LAN. Therefore the distributed system can be defined as:

"A distributed system is a collection of independent computers (nodes) that appears to its users as a single coherent system". [1]

Distributed file systems (DFS) are a part of distributed systems. DFS do not directly serve to data processing. They allow users to store and share data. They also allow users to work with these data as simply as if the data were stored on the user's own computer.

Over the past few years need of storing a huge amount of data has grown. Whether data are of multimedia types (e.g. images, audio, or video) or are produced by scientific computation, they should be stored for future reuse or for sharing among users. Users also need their data as quick as possible. Data files can be stored on a local file system or on a distributed file system. Local file system provides the data quickly but does not have enough capacity for storing a huge amount of the data. On the other hand, a distributed file system provides many advantages such as reliability, scalability, security, capacity, etc.

Distributed file systems provide persistent storage of unstructured data, which are organized in a hierarchical namespace of files that is shared among networked nodes. Files are explicitly created and they can survive the lifetime of processes and nodes until explicit deletion. As such they can be seen as the glue of a distributed computing infrastructure.

II. RELATED WORK

This section includes different types of distributed systems and different techniques for file searching. Types of distributed systems are *Distributed Computing Systems, Distributed Information Systems, Distributed Pervasive Systems. later* The types of file searching techniques are explained which are namely FusionFS, Apache Lucene, Grep, and Cloudera Search.

Types of distributed system are:

A. Distributed Computing Systems

- Used for high performance computing tasks
- Cluster computing systems
- Grid computing systems

B. Distributed Information Systems

- Systems mainly for management and integration of business functions
- Transaction processing systems
- Enterprise Application Integration

C. Distributed Pervasive Systems

- Mobile and embedded systems
- Home systems
- Sensor networks

There are various types of distributed systems, such as Clusters [10], Grids [11], P2P (Peer-to-Peer) networks [12], distributed storage systems and so on. A cluster is a dedicated group of interconnected computers that appears as a single super-computer, generally used in high performance scientific engineering and business applications. Grid enables coordinated sharing and grouping of distributed resources based on user's QoS (Quality of Service) requirements. Grids are usually used to support applications emerging in the areas of e-Science and e-Business. P2P networks are decentralized distributed systems providing

applications such as file-sharing, instant messaging, online multiuser gaming and content distribution over public networks. Distributed storage systems such as NFS (Network File System) provide users with a unified view of data stored on different file systems and computers which may be on the same or different networks.

File Searching Techniques:

A. FusionFS

FusionFS [3] is a distributed file system in High-speed Computing. It co-occur with current parallel file systems, optimized for both a subset of HPC and Many-Task Computing workloads. FusionFS is a user-level file system that runs on the compute resource infrastructure, and enables every compute node to actively participate in the metadata and data management. Distributed metadata management is implemented using ZHT [2], a zero-hop distributed hash table. ZHT was used for the specific requirements of high-end computing (e.g. trustworthy/reliable hardware, fast networks, non-existent "churn", low latencies, and scientific computing data-access patterns). The data is partitioned and spread out over many nodes based on the data access patterns. Replication is used to ensure data availability, and cooperative caching delivers high aggregate throughput.

B. Apache Lucene

Lucene [4] is a high performance, scalable Information Retrieval (IR) library. Information Retrieval [5] refers to the process of searching for documents, information within documents or metadata about documents. Lucene provides search capabilities to an application. It's a mature, free, open-source project implemented in Java. Lucene provides a powerful core API that requires minimal understanding of full-text indexing and searching.

Documents are atomic unit of indexing and searching. It can be thought as a box holding one or more Fields. These fields contain the content of the data. For each Field, it has a name identifier and series of detailed options that describe what Lucene should do with the Field's value when documents are added to the index. Raw content sources are first translated into Lucene's Documents and Fields before they are added to the index. During search, it is the values of these fields that are searched. When a field is indexed in Lucene, tokens are created from its text value using a process called tokenization. A Field's value can also be stored. When this happens an unanalyzed/ un-tokenized value (a copy of the value) is stored in the index so that it can be retrieved later.

Indexing text based files for searching is a very common practice, and there are utilities for searching for content in a file on a single machine like GNU Grep. However, there are very few distributed indexing and searching methods available. One common example of distributed search systems is Google [15]. These search engines are however primarily web based meaning their index is built using link crawling and aggregation, and requires huge processing power to build and keep up to date.

C. Cloudera Search

Cloudera Search [6], developed by Cloudera uses Apache Solr [7], an enterprise search version of Apache Lucene. Cloudera Search includes a highly scalable indexing workflow based on MapReduce. A MapReduce workflow is launched onto specified files or folders in HDFS, and the field extraction and Solr schema mapping is executed during the mapping phase. Depending on system configuration and preferences, Solr is used by reducers to write the data as a single index or as index shards. Once the indexes are stored in HDFS, they can be queried using standard Solr mechanisms. Apache Solr as mentioned earlier is an open source enterprise search platform built on Apache Lucene. It takes Lucene a step further by providing distributed indexing, replication and load-balanced querying, automated failover and recovery and centralized configuration.

D. Grep Search

Grep [8] searches input files for lines containing a match to a given pattern list. When it finds a match in a line, it copies the line to standard output (by default), or produces whatever other sort of output you have requested with options

III. Comparison: FusionFS Vs. Grep, and Cloudera Search

Here a brief overview on how the search mechanism works in Grep and Hadoop Grep, and how the indexing and searching mechanism works in Cloudera Search is given. Hadoop Grep [8] works different from the default UNIX Grep in that it doesn't display the complete matching line, but only the matching string

Cloudera Search provides the mechanism to batch index documents using MapReduce jobs. MapReduceIndexerTool [9] is a MapReduce batch job driver that takes a configuration file and creates a set of Solr index shards from a set of input files and the indexes are written into HDFS in a flexible, scalable, and fault-tolerant manner. The indexer creates an offline index on HDFS in the output directory. Solr merges the resulting offline index into the live running service. Comparision is based on following parameters:

A. Indexing Throughput

Indexing throughput is the size of file in MB we can index per second as the size increases. From Figure 1, on the 4 nodes, Cloudera's indexing mechanism does better than FusionFS. As the number of nodes is increased, FusionFS does much better than Cloudera Search by at least 2.5x. This is because as we scale the number of nodes, the workload on each node is reduced. This however doesn't happen in the case of Cloudera Search, increasing the number of nodes has little impact on the indexing throughput.

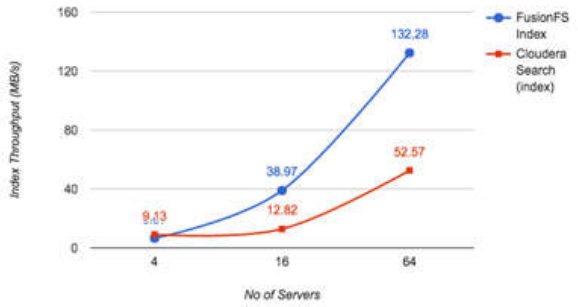

Figure 1: Indexing throughput in 10GB Cluster.

B. Search Latency

Search latency is the time it takes for the server to respond to a search request from the client. Since Grep is not a parallel tool, we played fair when comparing with the others as shown in Figure 2.

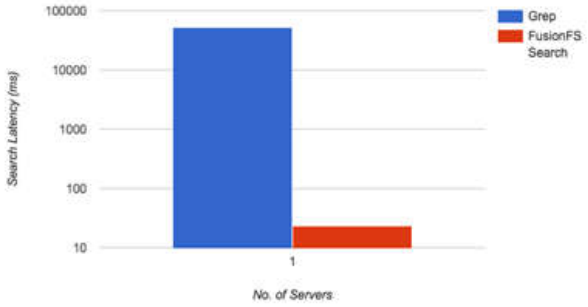

Figure 2: Search Latency of Grep and FusionFS in 1 node of 10GB

C. Scalability of Search

Scalability of search is measuring how much load one search server can handle; therefore all the nodes are considered acting as clients. Since client and server can co-exists on the same node without any interference, we also made one of the nodes a server. With all clients having the location of this search server, we made search requests and measured the search throughput. Figure 3 shows results on a 2.5GB dataset. The search throughput of Grep, FusionFS Search and Cloudera Search are compared making 4 concurrent client requests. Figure 3 shows that FusionFS Search responds to more queries per second than Cloudera Search and Grep. Scaling up, FusionFS Search will show to scale as the number of queries per second increases.

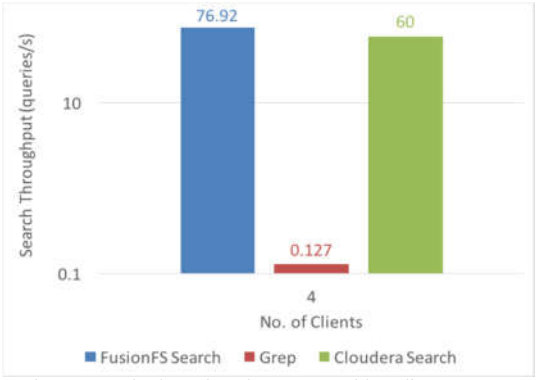

Figure 3: FusionFS Search Throughput in a 2.5GB with 4 client's concurrent request

IV. Existing Technologies

Over last decades a lot of research has been done in order to develop efficient searching mechanisms in large scale distributed systems. In this section we present basic and well-known search techniques that have been researched recently. Basic search approaches according to the network topologies are explained below.

A. Structured Peer-to-Peer Networks

Structured peer-to-peer networks are often exploiting distributed hash table mechanism (DHT). Distributed hash table is a system that provides effective localizations of data objects among the file-sharing network. It uses hash functions to

effectively map data-objects to keys, constructing the routing table with <key, value> pairs. The hash function is distributed all over the system hence each node can map data-objects to the keys equally. Therefore, every key has its specific place within the system. In case of dynamic node arrivals and departures, only small number of data objects have to be remapped causing minimal amount of system disruption. Query routing based on the DHT mechanism loses its semantic information. In addition, DHTs functionality was improved with their application to systems focused on range queries retrieval. However, these systems are more complicated and harder to maintain. DHTs are mainly used in structured environments.

B. Unstructured Peer-to-Peer Networks

In this section we introduce systems where nodes are not organized according to some structure. They do not distribute their data to the rest of the system according to the indexing rule. Therefore, nodes are not aware of the location of requested data-objects and use blind search mechanisms. Flooding and random walks are particularly exploited. The TTL mechanism is employed to avoid the infinite forwarding of the query or an occurrence of loops. The TTL is described as a limit that bounds the time to resolve the query. When the TTL value is achieved and the query is not resolved, searching is aborted without the result.

C. Centralized Peer-to-Peer Networks

The eDonkey [13] proposes searching in the centralized peer-to-peer network. Several servers are used and controlled by power users. They are able to communicate among themselves. Servers do not provide file-sharing. However, they provide information about shared files and their locations. Searching is process where a client browses server databases seeking for a required file. Thereafter, the server provides the location of client that maintains the queried file. The file-sharing is provided on the client level.

The Napster [14] is the file system used mainly for sharing music files among users. It consists of the central server (or servers) that maintain meta-data of currently connected client-users. The client forwards a query to the server, which decides if any other client currently connected to the system shares proposed data. The client receives respond and connects with given client (clients) to retrieve data. File-sharing is provided on the client level. The Napster is adjusted for dynamic environment.

D. Linear Search

Linear search or sequential search algorithm searches an element by examining each element in the list and comparing it with the search element from first element to the last element . It is called linear or sequential because this algorithm scans the elements in a list in linear manner or in sequence by scanning one by one element in a list.

If an element is found then index, flag signal or value can be returned or processed, otherwise special index as -1 or flag signal can be returned. Time required to search an element if exist or to determine that element is not found depends on the total number of elements in the list. So the time complexity of linear search is O(N)[16][17].

Worst case requires N comparison if an element is found at the last position N or an element does not exist in the list. Average time required by linear search on an average we can say an element may be at center of the list, so average time for this algorithm is O(N/2) which is again linear in nature.

The best case time is O(1) which is constant and element is found at the first position. Due to its simplest implementation it can be applied to array list as well as all types of linked lists. So it is easy to implement but it is not useful when the size of the list is too large. Because the time require is proportional to total number of elements N. So it is useful when we have small size of an array or a linked list but require more time when an array becomes large.

E. Binary search

Binary search algorithm finds the position of a specified input value (the search "key") within an array sorted by key value. The list should be arranged in ascending or descending order to perform searching using binary search. In each step, the algorithm compares the search key value with the key value of the middle element of the list. If the keys match, then a matching element has been found and its index, or position, is returned. Otherwise, if the search key is less than the middle element's key, then the algorithm repeats its action on the sub-array to the left of the middle element or, if the search key is greater, on the sub-array to the right. If the remaining list to be searched is empty, then the key cannot be found in the list and a special "not found" indication is returned. A binary search halves the number of items to check with each iteration, so locating an item (or determining its absence) takes logarithmic time.

Example: The list to be searched: L = 1 3 4 6 8 9 11. The value to be found: X = 4.
Compare X to 6. X is smaller. Repeat with L = 1 3 4.
Compare X to 3. X is bigger. Repeat with L = 4.
Compare X to 4. They are equal. Search is done, X is found.
This is called Binary Search: each iteration of (1)-(4) the length of the list we are looking in gets cut in half; therefore, the total number of iterations cannot be greater than logN.

V. System Design

System design totally describes the overall design and architecture of the system. This includes implemented techniques and their algorithms, etc. This dissertation work is about searching files in Distributed system, and investigating different techniques based on various parameters. It's design is best understood with the help of figure 4.

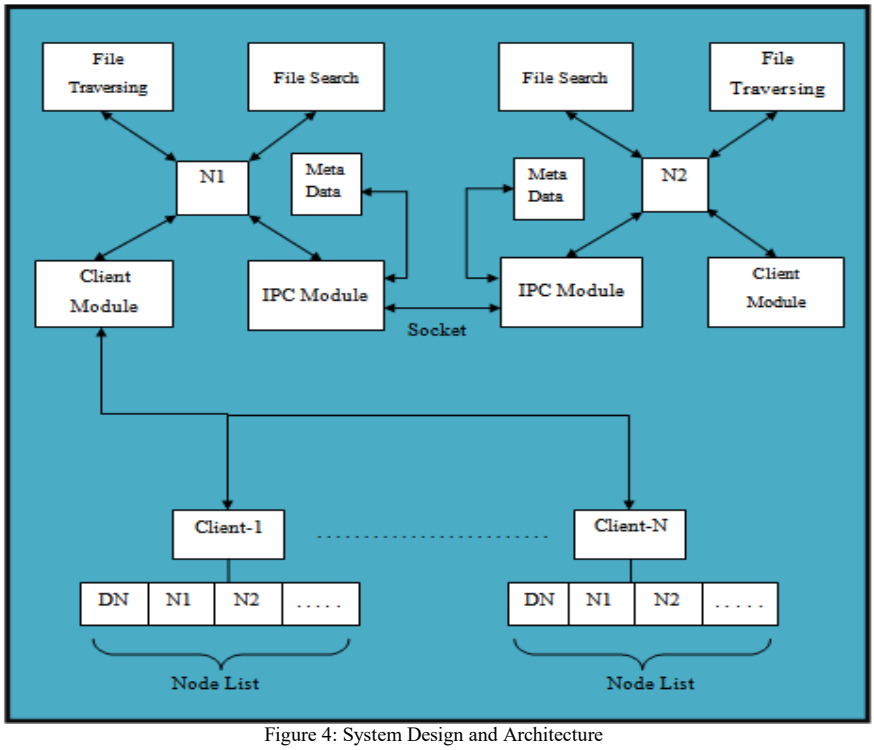

Figure 4: System Design and Architecture

A. Proposed System Architecture

The design of system shown in previous section with the help of figure 4, there are two server nodes represented by N1 and N2 in the middle of four modules namely Client module, IPC module, Searching module and Traversing module.

Function/working of each module is explained below:

1. IPC Module:

Inter-process communication (IPC) is a mechanism that allows the exchange of data between processes. By providing a user with a set of programming interfaces, IPC helps a programmer organize the activities among different processes. IPC allows one application to control another application, thereby enabling data sharing without interference.

IPC enables data communication by allowing processes to use segments, semaphores, and other methods to share memory and information. It facilitates efficient message transfer between processes. It has the capability of using publish/subscribe and client/server data-transfer paradigms while supporting a wide range of operating systems and languages. All these functions are handled by IPC module to accomplish the communication between processes and nodes.

2. Client Module:

Client module performs all the function that must be performed at the time when there's a request for any file. It has information regarding each node connected with it plus those nodes also have each other's files list. There can be more than one clients connected to the server and all client's working is handled by client module.

3. File Search:

This is a searching module for files in Distributed System. It can use any technique among Hashing, Binary search tree or Binary search. By selecting any one method, if the file is in server it directly returns to the client with file and its contents otherwise with the help of IPC module, it broadcasts that request to all nodes which will check which node might contain the requested file. Then return with the node (its IP address) having requested file (if that file exists on any node connected in DS) and finally returning the file to client.

4. File Traversing:

File traversing module is there to know about each and every files present on the node. It lists all the files with their subdirectories and path. So, when we want to know about which files are present on any node this module helps us for that. Client module is connected with appropriate clients which is in turn have number of nodes connected to it.

B. Proposed Algorithms

Here we discuss about the techniques or algorithms with the help of which we are performing the work of searching files in distributed systems. The existing techniques involved structured peer-to-peer network, unstructured peer-to-peer network, and centralized peer-to-peer network. These techniques however are more complicated and harder to maintain. Unstructured Peer-to-Peer Networks do not distribute their data to the rest of the system according. Therefore, nodes are not aware of the location of requested data-objects and use blind search mechanisms. The TTL mechanism is employed to avoid the

infinite forwarding of the query or an occurrence of loops. When the TTL value is achieved and the query is not resolved, searching is aborted without the result. Because of this abortion of file request many times the file is not returned even the file which exists on the system. Linear search is a simpler and better option but it doesn't perform well as number of elements to be searched from increases. Also its result is acquired much slower than needed in such environments. Binary search is a way better improvement over linear search still it doesn't provide the needed outputs in distributed environment. Hence this paper aims towards finding an effective and efficient file searching techniques in distributed systems. Proposed algorithms involve usage of Binary search trees, hashing and linear search. They are explained as below:

1. Binary Search Tree:

In a binary search tree each node have a left and right child. Any one child, or both children, may be missing. Figure 5 illustrates a binary search tree. Assuming k represents the value of a given node, then a binary search tree has the following property: all children to the left of the node have values smaller than k, and all children to the right of the node have values larger than k. The top of a tree is known as the root, and the exposed nodes at the bottom are known as leaves. In Figure 5, the root is node 20 and the leaves are nodes 4, 16, 37, and 43. The height of a tree is the length of the longest path from root to leaf. For this example the tree height is 2.

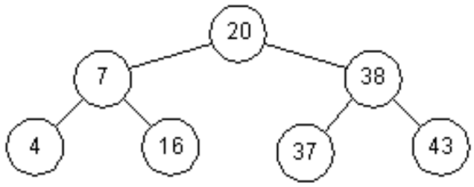

Figure 5: A Binary Search Tree

2. Hashing:

Hashing method is used to store data in an array so that sorting, searching, inserting and deleting data is fast. For this, every record needs unique key.

The Basic idea is not to search for the correct position of a record with comparison but to compute the position within the array. The function that returns the position is called 'Hash function' and the array is called the 'Hash table'.

VI. Conclusion

In this paper, we dealt with the issue of effective and efficient searching and traversing in distributed systems. One way of achieving better system performance is using a distributed system. The large amounts of data generated from high performance applications leads to problems when users need to locate these files for later use. The systems were divided to categories and the survey of distributed searching systems was introduced. It revealed the possibilities we could exploit to propose the new system.

Hence, the proposed system makes use of hashing indexing technique along with BST for searching in distributed systems. This proposed work tries to achieve scalability which describes the capability of a system, network, or process to handle a growing amount of work, with respect to search of files in distributed environment.

REFERENCES

[1] Andrew S. Tanenbaum and Maarten Van Steen, Distributed Systems : Principles and Paradigms. Upper Saddle River: Prentice Hall, 2002.

[2] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, 2013.

[3] Z. Dongfang, et al., "FusionFS: Toward supporting data- intensive scientific applications on extreme-scale distributed systems," in Proceedings of IEEE International Conference on Big Data, 2014.

[4] Apache. (2016). Apache Lucene [Online]. Available: http://lucene.apache.org/ core/.

[5] M. McCandless, E. Hatcher and O. Gospodnetic, Lucene in Action, 2nd. ed. Shelter Island, NY: Manning Publications Co, 2010.

[6] Cloudera. (2014). Cloudera Search [Online]. https://www.cloudera.com/ products/ apache- hadoop/apache-solr.html.

[7] Apache. (2016). Apache Solr [Online]. Available: http://lucene.apache.org/solr/.

[8] Apache. (2009, Sept). Hadoop Grep [Online]. Available: https://wiki.apache.org /hadoop/Grep.

[9] Cloudera. (2014). MapReduceIndexerTool [Online]. Available: http://www. cloudera.com/documentation/archive/search/1-3-0/Cloudera-Search-User-Guide/csug_ mapreduceindexertool.html.

[10] R. Buyya (editor), High Performance Cluster Computing, Prentice Hall, USA, 1999.

[11] I. Foster and C. Kesselman (editors), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.

[12] R. Subramanian and B. Goodman (editors), Peer-to-Peer Computing: Evolution of a Disruptive Technology, Idea Group Inc., Hershey, PA, USA, 2005.

[13] O. Heckmann, A. Bock, A. Mauthe, and R. Steinmetz, "The eDonkey File-Sharing Network," Multimedia Kommunikation (KOM), Technische Universität Darmstadt. 2000.

[14] The Napster website, [Online]. Available: http://www.napster.com.

[15] L. A. Barroso, J. Dean, U. Holzle, "Web search for a planet: The Google cluster architecture," in IEEE Micro, 2003, pp. 22–28.

[16] An Introduction to data structures with applications -(Mcgraw Hill Computer Science Series) [Jean- Paul Tremblay, Paul G. Sorenson, P. G. Sorenson].

[17] BINARY SEARCH ALGORITHM, Ancy Oommen , Chanchal Pal, 2014 IJIRT |Volume 1 Issue 5 | ISSN: 2349-6002.