International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Investigating File Searching Methods in Distributed Systems A Summary

Shahida Begum[a]

[a]*cse dept PDIT Hospet India*shahida@pdit.ac.in

DOI: https://doi.org/10.55248/gengpi.2022.3.8.23

A B S T R A C T

Large amounts of data are generated by high performance applications. The huge volume of data produced by computation causes data to be scattered throughout the file system. When users need to find these files for subsequent use, issues start to arise. This might not be a problem for modest numbers of files, but as the quantity and size of files rise, it becomes more challenging to locate particular files on the file system using standard techniques. The research and examination of file searching methods in distributed systems is the primary subject of this paper. Analyzes various distributed environments as well as current distributed system search techniques.

Keywords: File Searching Methods, Distributed Systems

1. Introduction

Computations today require powerful gear. A strategy for achieving outcomes more quickly is to keep buying new hardware. However, purchasing a supercomputer is an expensive option, and the software installation process for these modern supercomputers takes a long time. The costs associated with maintaining them are also substantial. The use of a distributed system is another method for improving system performance. Several computers are linked together in a dispersed system, typically by LAN. A distributed system is thus described as "a collection of independent computers (nodes) that appears to its users as a single coherent system." [1] Distributed systems include distributed file systems (DFS). Data processing is not directly served by DFS. Users can share and store data via them. Additionally, they enable users to interact with these data as easily as if they were kept on their own computer.

The requirement for massive amounts of data to be stored has increased over the last few years. Whether the data are from scientific calculation, multimedia (such as photographs, music, or video), or both, they should be stored for later use or dissemination among users. Additionally, users require their data as soon as possible. Both a distributed file system and a local file system can be used to store data files. Although the local file system supplies the data rapidly, it is not large enough to store a significant amount of data. However, a distributed file system has many benefits, including scalability, capacity, security, and reliability.

Unstructured data is arranged in a hierarchical namespace of files that is shared by networked nodes and is stored persistently by distributed file systems. Until they are expressly deleted, files can survive the lifetime of processes and nodes. They can thus be considered the foundation of a distributed computing architecture. [14]15][16][17] [18].

2. CONNECTED WORK

Different distributed system kinds and file searching methods are covered in this section. Distributed computing systems, distributed information systems, and distributed pervasive systems are examples of distributed systems. The several categories of file searching methods—FusionFS, Apache Lucene, Grep, and Cloudera Search—are described.

* *Corresponding author.*
E-mail address: *shahida@pdit.ac.in*

Various distributed system types include:

A. Systems for Distributed Computing

• Employed for activities requiring high performance computing

• Computerized clusters

Systems for grid computing

Systems for Distributed Information

• Mainly management and business function integration systems

• Systems for handling transactions

• Integration of business applications

Distributed omnipresent systems (C)

• Embedded and mobile systems

• Home appliances

Networks of sensors

Clusters [10], Grids [11], P2P (Peer-to-Peer) networks [12], distributed storage systems, and other forms of distributed systems are only a few examples. A cluster is a purpose-built collection of linked computers that works together to simulate a single supercomputer and is typically employed in high performance research, technical, and commercial applications. Based on user-specified QoS (Quality of Service) requirements, the grid enables coordinated sharing and grouping of distributed resources. The majority of the time, grids are utilized to enable new e-Science and e-Business applications. P2P networks are distributed, decentralized systems that provide functions like content distribution via open networks, instant messaging, and file-sharing. Users have a unified view of the data stored on many file systems and machines that may be connected to the same network or to other networks thanks to distributed storage systems like NFS (Network File System).

Techniques for searching files:

One. FusionFS

High-speed computing uses a distributed file system called FusionFS [3]. It co-occurs with contemporary parallel file systems that are tailored for a selection of both Many-Task Computing workloads and HPC workloads. The computing resource infrastructure uses FusionFS, a user-level file system that enables each compute node to actively participate in the metadata and data management. A zero-hop distributed hash table called ZHT [2] is used to implement distributed metadata management. For the particular needs of high-end computing (such as dependable hardware, quick networks, no "churn," low latencies, and scientific computing data-access patterns), ZHT was adopted. Based on the data access patterns, the data is divided and dispersed over numerous nodes. Data availability is ensured through replication, and high aggregate throughput is delivered by cooperative caching.

B. Lucene by Apache

A high-performance, scalable information retrieval (IR) library is Lucene [4]. Searching for papers, information within documents, or metadata about documents is referred to as information retrieval [5]. An application gets search capabilities from Lucene. It is an established, open-source Java project that is free to use. Lucene offers a robust core API that only necessitates a passing familiarity with full-text indexing and searching.

The building block of indexing and searching are documents. It might be compared to a box that contains one or more Fields. The data's substance is contained in these fields. Each Field is identified by a name and a number of specific parameters that specify what Lucene should do with the value of the Field when documents are added to the index. Before being included to the index, raw content sources are converted into Lucene's Documents and Fields. These fields' values are searched while performing a search. Tokenization is the process of turning a field's text value into a token when a field is indexed in Lucene. The value of a Field can also be saved. When this occurs, a copy of the value that hasn't been tokenized or processed is saved in the index so it may be accessed later.

It is very common practice to index text-based files for search purposes, and tools like GNU Grep can be used to search for content in a file on a single computer. Nevertheless, there aren't many distributed indexing and searching techniques out there. Google is a typical illustration of a distributed search system [15]. However, because they are mostly web-based, these search engines need a lot of processing power to construct and maintain their indexes, which are created by link crawling and aggregation.

Cloudera Search C

Apache Solr [7], an enterprise search variant of Apache Lucene, is used by Cloudera Search [6], a product of Cloudera. A MapReduce-based highly scalable indexing procedure is part of Cloudera Search. Launching a MapReduce workflow onto specific HDFS files or folders causes the field extraction and Solr schema mapping to be carried out. Reducers use Solr to determine whether to write the data as a single index or as index shards based on the system's settings and preferences. Utilizing typical Solr techniques, queries can be made once the indexes have been placed in HDFS. As was already noted, Apache Lucene serves as the foundation for Apache Solr, an open source enterprise search platform. By offering distributed indexing, replication, load-balanced querying, automated failover and recovery, and central configuration, it goes beyond Lucene.

Grep Search, D.

Grep [8] looks for lines in input files that match a specified pattern list. If a match is found in a line, it copies the line to standard output (by default) or generates any additional output you've specified using the options.

III. FusionFS vs. Grep, and Cloudera Search Comparison

Here is a brief explanation of how Grep and Hadoop Grep's search mechanisms, as well as Cloudera Search's indexing and searching mechanisms, function. Hadoop Grep [8] differs from the standard UNIX Grep in that it simply displays the matched string rather than the entire matching line.

Using MapReduce jobs, Cloudera Search offers the capability to batch index documents. With the help of a configuration file and a collection of input files, the MapReduceIndexerTool [9] produces a set of Solr index shards. The indexes are then flexible, scalable, and fault-tolerantly written into HDFS. In the output directory, the indexer builds an offline HDFS index. The generated offline index is combined by Solr with the currently active service. The criteria used for comparison are as follows:

Indexing Throughput, first

As the size of the file increases, our ability to index it is measured in megabytes (MB) per second. Figure 1 shows that Cloudera's indexing system outperforms FusionFS on the four nodes. As the number of nodes rises, FusionFS outperforms Cloudera Search by a factor of at least 2.5. This is because the demand on each node decreases as the number of nodes increases. However, this is not the case with Cloudera Search; adding nodes has little effect on the indexing throughput.

2. METHODOLOGY

The following describes how each module operates:

1. Inter-process communication (IPC) module: IPC is a method that enables data exchange between processes. IPC aids a programmer in organizing the activities among several processes by giving a user a set of programming interfaces. IPC makes it possible for one program to manage another, allowing for frictionless data interchange.

By allowing processes to share memory and information via segments, semaphores, and other mechanisms, IPC enables data communication. It makes message passing between processes more effective. It supports a large number of operating systems and languages and can use the publish/subscribe and client/server data-transfer paradigms. IPC module handles each of these tasks to facilitate communication between processes and nodes.

2. Client Module: The client module handles every task that needs to be completed when a request is made for any file. It contains details about each linked node, and those nodes also have lists of their shared files. There may be multiple clients connected to the server, and the client module manages all client activity.

3. File Search: This Distributed System module allows for file searching. Any method, including hashing, binary search trees, or binary search, can be used. By choosing one of the methods, if the file is on the server, it will immediately provide the file and its contents to the client; otherwise, it will broadcast the request to all nodes and those nodes will check to see if any of them might have the requested file. Finally, the file is returned to the client when the node (or its IP address) has received it (if the requested file is present on any linked nodes in DS).

4. File Traversing: The file traversing module allows you to view all of the files that are currently on the node. Each file is shown along with its path and subdirectories [19] [20] [21] [22]. Therefore, this module assists us when we want to discover which files are present on a particular node. The necessary clients are connected to the client module, which in turn has a number of nodes connected to it.

3.RESULTS:

.1. Binary Search Tree: A binary search tree has left and right children for each node. Both children or any one of them could be missing. Binary search trees are shown in Figure 5. A binary search tree has the property that all children to the left of the node have values smaller than k and all children to the right of the node have values bigger than k, assuming that k denotes the value of a particular node. The exposed nodes at the bottom of a tree are known as leaves, and the top of the tree is known as the root. The root is node 20 in Figure 5, while the leaves are nodes 4, 16, 37, and 43. The distance from root to leaf along the longest path determines a tree's height. The tree height in this illustration is 2.

a Binary Search Tree in Figure 5.

2. Hashing: Data is stored in an array using the hashing approach to make sorting, searching, inserting, and removing data quick and easy. Every record needs a different key for this.
The fundamental idea is to determine a record's position inside the array rather than using comparison to get its precise location. Hash function and Hash table are the names of the array and the function that returns the position, respectively.

4.CONCLUSION

1. Binary Search Tree: Each node in a binary search tree has both left and right children. Any one of the kids could be gone or even both of them. Figure 5 depicts binary search trees. Assuming that k represents the value of a certain node, a binary search tree has the feature that all children to its left have values lower than k and all children to its right have values larger than k. A tree's exposed nodes at the base are referred to as its leaves, while its exposed nodes at the top are referred to as its root. In Figure 5, node 20 represents the base, and nodes 4, 16, 37, and 43 represent the leaves. A tree's height is determined by the length of the longest path from root to leaf. In this example, the height of the tree is 2.

2. Hashing: Data is stored in an array using the hashing approach to facilitate sorting, searching, inserting, and removing data quick and simple. See Figure 5 for an example of a binary search tree. For this, a unique key is required for each record.
Instead of using comparison to identify a record's precise location inside the array, the basic idea is to ascertain that record's position within the array. The array's name and the function's name that returns the position are, respectively, Hash function and Hash table.[13] [14] 15] [16] [17] [18] .

REFERENCES

[1] Dr. C.N. Sakhale, D.M. Mate, Subhasis Saha, Tomar Dharmpal, Pranjit Kar, Arindam Sarkar, Rupam Choudhury, Shahil Kumar , "An Approach to Design of Child Saver Machine for Child Trapped in Borehole ", International Journal of Research in Mechanical Engineering, October-December, 2013, pp. 26-38.

[2] K. Saran, S. Vignesh, Marlon Jones Louis have discussed about the project is to design and construct a "Bore-well rescue robot" (i.e. to rescue a trapped baby from bore well), International Journal of Research in Aeronautical and Mechanical Engineering, Boar well rescue robot , pp. 20-30 April 2014

[3] G. Nithin, G. Gowtham, G. Venkatachalam and S. Narayanan, School of Mechanical Building Sciences, VIT University, India, Design and Simulation of Bore well rescue robot– Advanced, ARPN Journal of Engineering and Applied Sciences, pp. MAY 2014.

[4] Camera - Direct web search on google.com

[5] J. Burke and R.R.Murphy, "Human-robot interaction in USAR technical search: Two heads are better than one,"inProc.IEEE Int. Workshop ROMAN, Kurashiki, Japan, 2004, pp. 307-312.

[6] J. Casper and R. R. Murphy, "Human-robot interactions during the robot assisted urban search and rescue response at the world trade center," IEEE Trans. Syst., Man, Cybern. B, Cybern., Vol. 33, no. 3, pp. 367–385, Jun. 2013.

[7] R. R. Murphy, "Activities of the rescue robots at the World Trade Center from 11–21 September 2001," in Proc. IEEE Robot. Autom. Mag., 2004, pp. 50–61.

[8] Rodriguez, K. M., Reddy, R. S., Barreiros, A. Q., & Zehtab, M. (2012, June). Optimizing Program Operations: Creating a Web-Based Application to Assign and Monitor Patient Outcomes, Educator Productivity and Service Reimbursement. In DIABETES (Vol. 61, pp. A631-A631). 1701 N BEAUREGARD ST, ALEXANDRIA, VA 22311-1717 USA: AMER DIABETES ASSOC.

[9] Kwon, D., Reddy, R., & Reis, I. M. (2021). ABCMETAapp: R shiny application for simulation-based estimation of mean and standard deviation for meta-analysis via approximate Bayesian computation. Research synthesis methods, 12(6), 842–848. https://doi.org/10.1002/jrsm.1505

[10] Reddy, H. B. S., Reddy, R. R. S., Jonnalagadda, R., Singh, P., & Gogineni, A. (2022). Usability Evaluation of an Unpopular Restaurant Recommender Web Application Zomato. Asian Journal of Research in Computer Science, 13(4), 12-33.

[11] Reddy, H. B. S., Reddy, R. R. S., Jonnalagadda, R., Singh, P., & Gogineni, A. (2022). Analysis of the Unexplored Security Issues Common to All Types of NoSQL Databases. Asian Journal of Research in Computer Science, 14(1), 1-12.

[12] Singh, P., Williams, K., Jonnalagadda, R., Gogineni, A., &; Reddy, R. R. (2022). International students: What's missing and what matters. Open Journal of Social Sciences, 10(02),

[13] Jonnalagadda, R., Singh, P., Gogineni, A., Reddy, R. R., & Reddy, H. B. (2022). Developing, implementing and evaluating training for online graduate teaching assistants based on Addie Model. Asian Journal of Education and Social Studies, 1-10.

[14] Sarmiento, J. M., Gogineni, A., Bernstein, J. N., Lee, C., Lineen, E. B., Pust, G. D., & Byers, P. M. (2020).Alcohol/illicit substance use in fatal motorcycle crashes. Journal of surgical research, 256, 243-250.

[15] Brown, M. E., Rizzuto, T., & Singh, P. (2019). Strategic compatibility, collaboration and collective impact for community change. Leadership & Organization Development Journal.

[16] Sprague-Jones, J., Singh, P., Rousseau, M., Counts, J., & Firman, C. (2020). The Protective Factors Survey: Establishing validity and reliability of a self-report measure of protective factors against child maltreatment. Children and Youth Services Review, 111, 104868.

[17] Reddy Sadashiva Reddy, R., Reis, I. M., & Kwon, D. (2020). ABCMETAapp: R Shiny Application for Simulation-based Estimation of Mean and Standard Deviation for Meta-analysis via Approximate Bayesian Computation (ABC). arXiv e-prints, arXiv-2004.

[18] Reddy, H. B., Reddy, R. R., & Jonnalagadda, R. (2022). A proposal: Human factors related to the user acceptance behavior in adapting to new technologies or new user experience. International Journal of Research Publication and Reviews, 121-125. doi:10.55248/gengpi.2022.3.8.1

[19] Reddy, H. B. S., Reddy, R. R. S., & Jonnalagadda, R. (2022). Literature Review Process: Measuring the Effective Usage of Knowledge Management Systems in Customer Support Organizations. In International Journal of Research Publication and Reviews (pp. 3991–4009). https://doi.org/10.55248/gengpi.2022.3.7.45

[20] Reddy, R. R. S., & Reddy, H. B. S. (2022). A Proposal: Web attacks and Webmaster's Education Co-Relation. In International Journal of Research Publication and Reviews (pp. 3978–3981). https://doi.org/10.55248/gengpi.2022.3.7.42

[21] Reddy, H. B. S. (2022). A Proposal: For Emerging Gaps in Finding Firm Solutions for Cross Site Scripting Attacks on Web Applications. In International Journal of Research Publication and Reviews (pp. 3982–3985). https://doi.org/10.55248/gengpi.2022.3.7.43

[22] Lu, N., Butler, C. C., Gogineni, A., Sarmiento, J. M., Lineen, E. B., Yeh, D. D., Babu, M., & Byers, P. M. (2020). Redefining Preventable Death—Potentially Survivable Motorcycle Scene Fatalities as a New Frontier. In Journal of Surgical Research (Vol. 256, pp. 70–75). Elsevier BV. https://doi.org/10.1016/j.jss.2020.06.014