DOI: 10.18254/S278229070018060-2 Original article

Review of Metadata Storing and Searching Methods in DLT and Distributed Files Systems

© 2022 Obadah Hammoud [a]

[a] Department of Engineering Cybernetics,
National University of ScienceTechnology "MISiS", Moscow, Russia
Department of Information Technology, Tartous University,
Tartus, Syria
obadah.hammoud@gmail.com

Abstract. In this review the author explores currently available methods for storing and looking up files by metadata. Different available solutions were reviewed and their shortcomings have been highlighted. The purpose of this study is to evaluate existing metadata processing methods in order to provide a reliable basis for building a metadata indexing system for a distributed documents exchange system. Different projects and researches were considered: Luster, WekaFS, Ceph, Gluster, IPFS, HDFS, Spyglass, Smartstore, and Distributed Metadata Search for the Cloud research. It was concluded that all of the solutions reviewed did not meet the requirements for document metadata processing. In the end, the characteristics of the ideal system for the job were described.

Keywords: DLT, metadata search, e-document, B2B exchange

For citation: Hammoud, Obadah. 2022. Review of metadata storing and searching methods in DLT and distributed files systems. *Law & Digital Technologies* 2(1): 35–42.

INTRODUCTION

E-documents form an essential part of daily B2B data exchanges on the Internet. This includes contracts, service notes, official mails, financial documents and other types of e-documents. These documents have a different structure than random files (Sawadogo et al. 2019). They may contain specific metadata entries, like an author's name, account number, registration date, etc. Various standards can be used to represent these metadata entries, like MARC21, Dublin Core, ANSI ASC X12 and others. As these documents have their own structure, B2B services are required to process~~es~~ these documents based on various attributes.

Large corporations build their own services to manage e-documents using centralized solutions. As centralized solutions can be easily managed and deployed, they are more vulnerable to attacks and threats, which leads to critical consequences for the entire system (Golosova and Romanovs, 2018). A distributed architecture can help to avoid these threats. A recent trend in B2B systems is taking advantage of distributed architectures, such as immutability data and transparency of data processing and logs. Thereby, such well-known distributed technologies as DLT and IPFS are considered. The architecture should be designed to fit the requirements of processing e-documents, such as searching for different e-documents using several attributes.

The aim of this review is to study the most well-known projects which handle files in distributed systems, or metadata in general. The paper discusses how much these projects can be integrated into a distributed structure to store e-documents. For this purpose, the author highlights the flaws and advantages of each project.

1. REVIEW METHODOLOGY

Different technologies and solutions were considered in this study. These solutions were classified into three main categories:

1. Projects and research articles about storing data in distributed architectures, like Luster, IPFS, etc.
2. Projects and articles about methods to store metadata which handles a big number of files.
3. Projects and articles about storing metadata in blockchain.

This study utilized different open resources: research articles and white papers available on the Internet. Among the sources examined in the present review, there were developing projects mentioned and cited for the last 3–4 years, technical documentations which had been updated over the last year, and around which a community of developers was being formed.

The system for the task of processing e-documents metadata can be described by the following requirements:

1. Support for several attributes, with various data types like string, number, dictionary, etc.

2. The ability to work on several nodes. As there can be several nodes, having a single node to handle searching processes can lead to a bottleneck. Also, the system should be flexible when adding a new node, so the workload can be shared with new nodes without interference.

3. Support for searching data entries based on multiple attributes. It includes searching by a single attribute, several attributes, or even all attributes. The solution should be efficient in terms of speed and memory (for example, storing many trees is not efficient in terms of memory).

4. Support for complex queries. There are a number of situations where complex queries can be useful, such as finding data quickly or making reports. Complex queries may combine multiple attributes, such as intersections and aggregations.

5. Support for scheme changes for metadata. Eventually, there may be a need to upgrade the data schema, and the process should be feasible.

6. A method or protocol to agree on data must be implemented for nodes to resolve conflicts automatically. It is also necessary to synchronize the data between nodes to quickly add changes.

2. DISTRIBUTED FILESYSTEMS

2.1. Luster

Luster (Cluster File Systems 2002) is a clusters-based opensource filesystem, which works in distributed structures. Most HPFS (High Performance File System) use the Luster file system (Salunkhe 2016). The Luster file system mainly consists of three types of nodes (Fig. 1):

a. The client which uses the filesystem

b. Metadata server (MDS)

c. Object storage server (OSS)

The Luster filesystem can contain more than one metadata server, and these servers store data on devices called MDT (Metadata Targets). Also, ~~a~~ the Luster filesystem can contain one or more MDTs. MDTs contain all required metadata, including filenames, their permissions, folders, and so one.

These servers manage storage devices called OSTs (Object Storage Target). The Luster filesystem can be comprised of one or several OSSs, and each OSS can have a single or multiple OSTs. The total storage capacity of the system is the sum of the storage spaces of all OSTs.

MDS uses both round-robin and a weighted random algorithm to allocate OSTs' objects as follows:

When the free space variation among all the OSTs is less than a certain threshold (17%), MDS uses round-robin to choose the next OST to write the strip of data to. This case is called a balanced state.

If the free space variation among all the OSTs is more than the certain threshold (imbalanced state), it uses weighted random algorithm to allocate the next object. By using this algorithm, the file system can return to the balanced state by writing more objects to the OSTs that have more free space. The I/O performance decreases while the system is imbalanced.

As long as there might be more than one MDT, Luster uses DNE (Distributed Namespace) remote directories to assign each MDT a part of the overall MDT data. This process is called a DNE phase 1. MDTs can have nested relations, so one MDT can present a part of another MDT, but this can cause problems, because if a MDT is damaged, all the nested MDTs become useless.

The metadata of a single directory can be stored on several MDTs, where each MDT stores a part of this metadata. As an example, this directory contains several files, and multiple MDTs store the metadata of the files which are located within this directory, where each MDT stores a part of the files' metadata. Nevertheless, a single file has all its metadata stored on the same MDT.

The MDS servers are resource-intensive. According to the official documentation (Lustre* 2017), it is customary to allocate 16 cores and 256GB RAM for MDS servers. MDSs use Inode bit locks (Inode is a data structure used in Unix-like systems) for file lookup (González-Domínguez 2019). Inode bit locks function as follows: whenever a file is requested (for example, files/file1), the Inode structure is checked to see the location of the folder "files", for example it can be located in block "B1". Block "B1" is then checked to read the folder content and

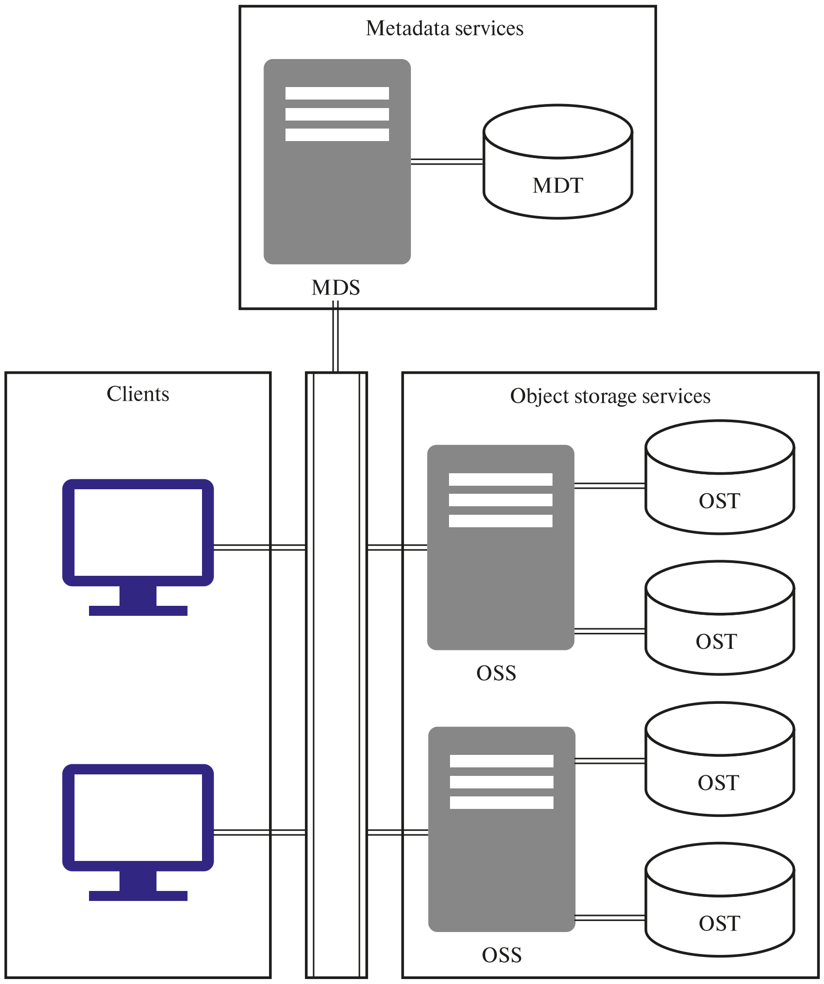

Figure 1: Luster storage architecture

the files location which are inside. As an example, the metadata of "File 1" can be located in block "B2". Finally, Block "B2" is checked, and it shows that the actual data is stored in block "B3".

Luster is not suitable for processing e-documents because:

1. This filesystem supports finding data by basic file attributes. The project is not designed to search for e-documents by their metadata, such as author, keywords, etc.

2. To store metadata, a central server with high specifications is required, which is considered a bottleneck.

2.2. WekaFS

Another filesystem designed to work on distributed structures is WekaFS (Benet 2014). WekaFS supports high bandwidth operations as it was originally designed for NVMEeoF. Also, WekaFS has a high scalability. Unlike Luster, WekaFS distributes the metadata across all the storage nodes (while other distributed filesystems like Luster use dedicated servers to host metadata). As a result, there will be no bottleneck point, as metadata servers affect the overall system performance. The metadata in WekaFS is distributed equally among the nodes, so no node is considered more important than another. The method of implementing metadata processes is patented. So, based on that:

1. WekaFS splits metadata across several servers, so there is not a bottleneck.

2. The metadata processing algorithm is patented, so it cannot be used.

2.3. IPFS, IPFS-Search

In the last few years, IPFS (Lustre Wiki Links n.d.) has received much attention. It is a P2P network which has its own protocol for sharing data in a distributed structure. IPFS can store different types of data, so storing NFT related data became easier with IPFS (Das et al. 2021; Wang et al. 2021).

IPFS can store metadata about NFT as a JSON object. An example would be: *{"name": "test" "description": "image description" "image": "ipfs:://jgifmrofskfjgm3k9g4mc4nfo5mv34s4mnf4mf43nf3erkjvvjkjk3f3cdg/test.png"}*

IPFS-Search (IPSE TEAM 2019) is a project which allows searching for IPFS content using metadata. This project uses IPFS-Tika to extract the metadata, and ElasticSearch 7 for searching process, which is a search engine used for indexing and searching purposes. So, IPFS does not have an embedded metadata searching algorithm. Some specific solutions have their own implementations, such as searching for NFT using IPFS-search.

2.4. Ceph

Ceph (Hilmi 2019) is a platform which uses object storage, and has its own filesystem (cephFS). It has similarities with Luster, as it also uses metadata servers (ceph-mds).

In cephFs, one of the metadata servers should be in active mode, while the rest of metadata servers should be in standby mode. If the active server fails or stops working, one of the remining servers becomes active to guarantee high availability. So, the disadvantages of using this project to process metadata are:

A central server with high specifications is required to store metadata, which is considered as a bottleneck

The project is not designed to search for e-documents by their metadata, such as author, keywords, etc.

2.5. Gluster

Gluster (Selvaganesan and Liazudeen 2016) is the combination of GNU and Luster, it is not based on Luster. Unlike Luster, Gluster does not have any metadata servers, and unlike WekaFS metadata is not required at all. Files are located using an elastic hashing algorithm. Due to this algorithm, it is possible to find the location of data fragments using calculations without having to search the indexes. A virtual volume can be located in several volumes (Fig. 2).

Gluster clearly does not meet the requirements to process e-documents:

1. The metadata is not stored at all, as the file location can be found by algorithms dynamically.
2. The project is not designed to search for e-documents by their metadata, such as author, keywords, etc.

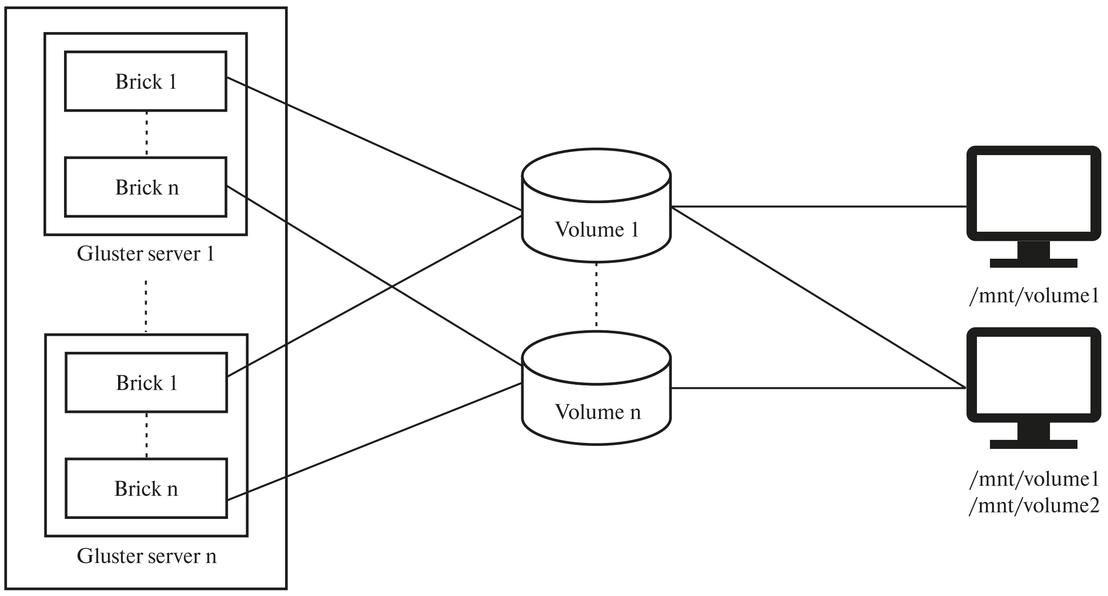

Figure 2: Gluster architecture

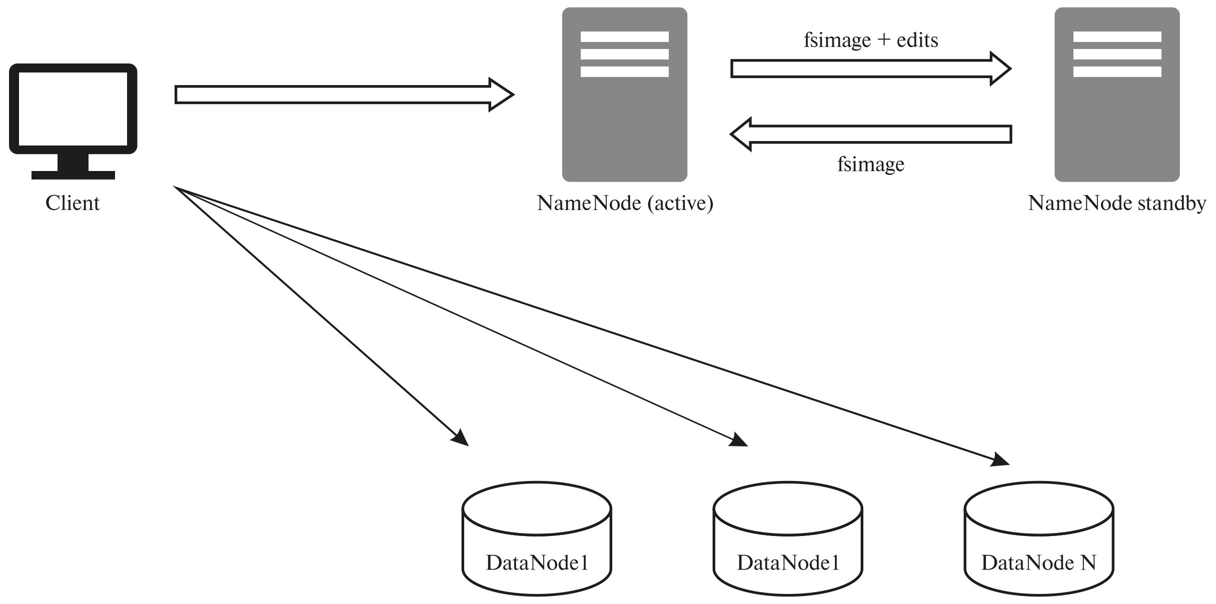

Figure 3: HDFS architecture

2.6. HDFS (Hadoop)

HDFS (Wang and Su 2013) is the filesystem used by Hadoop. In this storage platform, the files are split into blocks, and replicated on three servers; so if a server fails, the content is still accessible from the other two servers. HDFS has different node types (Fig. 3):

1. Master Nodes: Servers used for distributed storage management. They are called NameNodes. A NameNode holds the metadata about the cluster. There can be one active NameNode, and standby name nodes, so if the NameNode fails, one of the remaining NameNodes becomes active.

2. Worker Nodes: nodes where data is actually stored. They are also called DataNodes.

To detect nodes failures, a heartbeat message is sent every three seconds to the NameNode. When a DataNode stops sending heartbeat messages, the NameNode server assumes that this DataNode has failed, and it unmaps it from the cluster, so it is excluded from future writings / readings. Once the NameNode server receives the heartbeat message from the excluded DataNode again, the DataNode joins the cluster automatically.

This method has two main issues:

1. A central server with high specifications is required to store metadata, which is considered as a bottleneck

2. The project is not designed to search for e-documents by documents' metadata, such as author, keywords, etc.

3. METADATA PROCESSING PROGECTS

3.1. Spyglass

The Spyglass (Leung et al. 2009) project aims to speed up the search of files by their metadata in large storage systems. Metadata can be inode metadata, like file name, permissions, etc. or extra metadata, like a title or the author of a document. File metadata with close namespaces are grouped together in Spyglass. It is because the user tends to search for files in specific directories. As a result, only a subtree (a part of the whole tree) will be considered instead of searching the whole tree, which increases the search speed, while many other used solutions like DBMS treat namespaces as strings, so the system has to search all the available entries. In Spyglass, these partitions (subtrees) are required to be small, around 100,000 files per partition. Furthemore, a cache about these partitions is saved in memory, and is updated using the LRU algorithm. This project uses K-D trees to search partition indexes. Also, it uses index versioning to limit the update processes.

An important method for searching for data by attributes is the K-D tree. This tree splits data based on the mean, until the target data entry is found. Thus, when searching for an entry, we check its location, as data is split into 2 halves based on the mean, and the quarter (half of the half) is checked, and so on. The K-D tree supports multi-attributes, by splitting the data by the mean according to each dimension for each resulting part. In this way, the data is split according to the mean value of the x axis, for example, and the resulting collection of data is split according to the mean value of the y axis of this particular collection, and so on. The problem of this tree is that all attributes are considered in the selection process, and it is impossible to search for data by a subset of the attributes without building a new tree with this subset of attributes.

Steps we can take include:

The project mainly groups entries which are in close namespaces to increase search speed. When dealing with a system to store e-documents, the division of namespaces will not be required for search processes.

It uses K-D trees, which can be problematic when updating/inserting data frequently.

3.2. Smartstore

The Smartstore (Hua 2009) project uses R-Trees instead of K-D trees. In this project, the same metadata can contain several R-Trees to match different patterns. When the user sends a search query, Smartstore uses Bloom filter, or Minimum Bounding Rectangles for complex queries to determine if a server belongs to the query. Based on that:

This project uses R-trees, which handles multiple attributes well.

Search speed degrades by time when the attributes become overlapped (Yu 2016).

3.3. Distributed Metadata Search for the Cloud research

The "Distributed Metadata Search for the Cloud research" research (Yu 2016) suggests increasing the search speed by dividing metadata across multiple servers, so each server searches in a part of the indexes list. It recommends using MangoDB to store the indexes among the servers. On each server, the data is split again into several blocks, and each block is handled as a separate K-D-B tree. According to the research, it has to do with the fact that when the user searches for a query, several partitions can be excluded completely, and time is saved by not searching these blocks. When the user submits a search query, a dedicated server, called the organizer, determines which servers are relevant to the query, and after that, each server of the selected servers decides which blocks are relevant to the query. However:

1. The suggested study uses a centralized server.
2. This study is yet not deployed by any of well-known projects.

4. BLOCKCHAIN AND METADATA

Indexing is not a capability which is directly supported by current public blockchains. In Ethereum, if the functionality of indexing metadata is required, the user should write a smart contract with this functionality. In the research (Barriocanal et al. 2017) a structure for storing metadata on Ethereum is proposed. However, this study suggests a way to store only basic metadata on blockchains. Even if this study is modified to store other types of metadata, no other algorithms are discussed to increase the speed of search processes by attributes.

There is practically no information on how to search for metadata in DLT. BigchainDB, for instance, is a blockchain based database, which supports MangoDB, a document-oriented database. In Bigchain, metadata adds data to transactions, and it can be updated / added for each transaction. The release of BigchainDB V1.3 allows queries on the metadata, but technical implementation details are missing.

CONCLUSION

In this review, we discussed technologies used to store files in distributed systems, as well as the way how metadata is processed, having in mind that the purpose is to find a solution that suits the requirements of processing e-documents.

- Based on the results of the review, several functions necessary for the implementation of a full-fledged B2B exchange system were identified:
- Basic metadata: includes metadata related to normal files, such as data creation, file name, etc.
- Extended metadata: includes possible additional metadata, such as DOI, author name, etc.
- Search by metadata: includes searching for e-documents using different metadata attributes.

Project	Basic metadata	Extended metadata	Search by metadata	Complex requests	Distributed metadata nodes
Luster	Yes	No	Yes	No	No
WekaFS	Yes	No	Yes	No	Yes
Ceph	Yes	No	Yes	No	No
Gluster	No	No	No	No	No metadata servers
IPFS*	No	No	No	No	No
HDFS	Yes	No	Yes	Yes	No
Spyglass	Yes	No	Yes	No	No
Smartstore	Yes	No	Yes	Yes	No
Distributed Metadata Search for the Cloud research"	Yes	No	Yes	No	No

Table 1. Summary of the review

- Complex requests: an example would be finding all documents whose author is X in the period [Y-Z]
- Distributed metadata nodes: This is required to avoid having a bottleneck in the system

Table 1 presents the summary of our review.

IPFS originally doesn't process metadata. However, IPFS can be used with NFT, but metadata can be saved on nodes like in torrent, and this is not related to the purpose described above. We did not review CDN (BigchainDB GmbH 2018) as it is a technology to cache data by using servers in different locations for faster delivery of data content.

As evident, there is no ready solution for storing and processing e-documents' metadata, which satisfies the conditions outlined above, and which is necessary when dealing with different e-documents processing operations.

There should be no reliance on a single node, and the system should allow to search by a variety of attributes, whether by a singular attribute or by a combination of them in a fair speed.

E-documents contain metadata attributes which random files do not have. With a huge number of e-documents to store, the system must be able to locate desired documents using these attributes in a resource-efficient and speed-efficient manner.

REFERENCES

1. Barriocanal, Elena García, Salvador Sánchez-Alonso, and Miguel-Ángel Sicilia. 2017. "Deploying Metadata on Blockchain Technologies". In *Metadata and Semantic Research*, edited by Emmanouel GaroufallouSirje VirkusRania SiatriDamiana Koutsomiha. Communications in Computer and Information Science. Springer, Cham. https://doi.org/10.1007/978-3-319-70863-8_4
2. Benet, Juan. 2014. "IPFS – Content Addressed, Versioned, P2P File System." https://arxiv.org/abs/1407.3561
3. BigchainDB GmbH. 2018. "BigchainDB2.0 the Blockchain Database." https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
4. Cluster File Systems, Inc. 2002. "Lustre: A Scalable, High-Performance File System." https://cse.buffalo.edu/faculty/tkosar/cse710/papers/lustre-whitepaper.pdf
5. Das, Dipanjan, Priyanka Bose, Nicola Ruaro, Christopher Kruegel, and Giovanni Vigna. 2021. "Understanding Security Issues in the NFT Ecosystem." ArXiv:2111.08893. https://arxiv.org/abs/2111.08893.
6. Golosova, Julija, and Andrejs Romanovs. 2018. "The Advantages and Disadvantages of the Blockchain Technology." IEEE6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE). https://doi.org/10.1109/aieee.2018.8592253
7. González-Domínguez, Jorge, Verónica Bolón-Canedo, Borja Freire, and Juan Touriño. 2019. Parallel Feature Selection for Distributed-Memory Clusters. *Information Sciences* 496: 399—409. https://doi.org/10.1016/j.ins.2019.01.050.
8. Hilmi, Muhammad, Eueung Mulyana, Hendrawan Hendrawan, and Adrie Taniwidjaja. "Analysis of Network Capacity Effect on Ceph Based Cloud Storage Performance." 2019 IEEE13th International Conference on Telecommunication Systems, Services, and Applications (TSSA), 2019. https://doi.org/10.1109/tssa48701.2019.8985455.
9. Hua, Yu, Hong Jiang, Yifeng Zhu, Dan Feng and Lei Tian. 2009. "SmartStore: a new metadata organization paradigm with semantic-awareness for next-generation file systems." In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*: 1—12.

10. IPSE TEAM. 2019 "IPSE: A Search Engine Based on IPFS IPSE TEAM." https://ipfssearch.io/IPSE-whitepaper-en.pdf

11. Leung, Andrew W., Minglong Shao, Timothy Bisson, Shankar Pasupathy and E.L. Miller. 2009 "Spyglass: Fast, Scalable Metadata Search for Large-Scale Storage Systems." In *Proceedings of the 7th USENIX Conference on File and Storage Technologies*, San Francisco, CA, February 2009.

12. Lustre* systems and network administration. 2017. "Introduction to Lustre* Architecture". October 2017. https://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf

13. Lustre Wiki Links.n.d. Lustre Metadata Service (MDS). https://wiki.lustre.org/Lustre_Metadata_Service_(MDS)

14. Salunkhe, Rushikesh, Aniket D Kadam, Naveenkumar Jayakumar, and Shashank Joshi. 2016. "Luster a Scalable Architecture File System: A Research Implementation on Active Storage Array Framework with Luster File System." *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016. https://doi.org/10.1109/iceeot.2016.7754852.

15. Sawadogo, Pegdwendé, Tokio Kibata, and Jérôme Darmont. 2019 "Metadata Management for Textual Documents in Data Lakes." In *Proceedings of the 21st International Conference on Enterprise Information Systems* 1: ICEIS, 72–83, 2019, Heraklion, Crete. https://doi.org/10.5220/0007706300720083.

16. Selvaganesan, Manikandan, and Mohamed Ashiq Liazudeen. 2016. "An Insight about Glusterfs and Its Enforcement Techniques." *International Conference on Cloud Computing Research and Innovations (ICCCRI),* 2016. https://doi.org/10.1109/icccri.2016.26

17. Wang, Xin, and Jianhua Su. 2013. "Research of Distributed Data Store Based on HDFS." *International Conference on Computational and Information Sciences*, 2013. https://doi.org/10.1109/iccis.2013.384.

18. Wang, Qin, Rujia Li, Qi Wang, and Shiping Chen. 2021. "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges." ArXiv:2105.07447 [Cs], May. https://arxiv.org/abs/2105.07447.

19. Yu, Yang, Yongqing Zhu and Juniarto Samsudin. 2016. Distributed Metadata Search for the Cloud. *Journal of Communications* 11(1): 100–107.