International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 11 Issue 4 ǁ April 2023 ǁ PP. 182-187

Effective way of optimizing memory by self-adjusting resource al location in Hadoop environment

Prof. Shwetha K S | Pooja Adiveppa Karagi | Pramod N Melmari | Siddharth Rao | Godlee CJ

Computer Science and engineering
Dayananda Sagar College of Engineering
Bangalore-560078

Abstract:*The Hadoop file system is designed to handle large file volumes and is fault tolerant. It employs a master/slave architecture in which there is no shared data between the master and slave. The master, also known as the name node server, stores metadata in its own storage. However, managing a large number of small files can be challenging for the name node and can negatively impact the performance of the data nodes, making the file system memory intensive. To avoid this issue, HDFS does not serve prefetched or bundled files, and therefore, its I/O speed is improved.*
To enhance the efficiency of storing and accessing small files in the file system, the Extended Hadoop Distributed File System (EHDFS) was introduced. This approach involves storing the files in a combined file on the data node or client, called the constituent file map. An indexing mechanism is used to access individual files from the combined file, and index prefetching is used to reduce the name node loadand improve I/O performance. As a result, the name node footprint is reduced, making the file system more efficient.
Keywords*-Hadoop, Name Node, Data node, HDFS,EHDFS,ConstituentFile Map.*

Date of Submission: 01-04-2023 Date of acceptance: 12-04-2023

I. Introduction:

When files in HDFS are significantly smaller than the default block size of 64MB, they are referred to as "tiny files." If you store a lot of small files, it can cause issues because HDFS cannot efficiently manage a large number of files. This is because each file, directory, and block in HDFS is represented as an in-memory object in a name node, which consumes about 150 bytes of storage. As a result, storing 10 million files that use blocks will consume approximately three gigabytes of storage. However, scaling much higher than this level is difficult with current hardware.

Storing small files results in inefficient data access patterns because it requires many seeks and hops from data node to data node to retrieve each small file. Additionally, when using map tasks, processing very small and numerous files can result in each map task processing very little input, leading to many more map tasks and additional accounting overhead. For example, splitting a 1GB file into 16 64MB chunks and comparing it to 10,000 100KB files, each using one mapping, shows that the job times can be tens or hundreds of times slower than with a single input file.

To avoid these issues, you can reuse task JVMs to run multiple map tasks in one JVM and avoid JVM startup overhead. It's also worth noting that HDFS is primarily intended for streaming access to large files.

II. LiteratureSurvey:

Liu Jiang, Bing Li ,Meina Song[1]: The HADOOP framework is a distributed file system that operates on top of HDFS and is adept at efficiently managing enormous amounts of data across vast clusters. It has gained widespread use in developing large-scale, high-performance systems. However, HDFS is not optimized for handling numerous small files, and performance issues can emerge. In response, many enterprises are now turning to cloud storage solutions, such as Amazon's S3, to store their data. As the Internet continues to advance, users may increasingly prefer to store their data and programs on cloud computing platforms. Nevertheless, since most personal data comprises small files, HDFS may not be suitable for fulfilling this need.

XianzhenRen,XiuhuaGeng,YiZhu[2]: Files that are much smaller than the default HDFS block size (64MB, or 128MB on newer Hadoop setups) are considered small files. While HDFS is primarily meant for storing large files, the nature of input sources and recording techniques often requires the storage of numerous small files in HDFS. However, HDFS is not optimized for managing small files as it is designed to process large amounts of data sequentially. As a result, managing small files in HDFS can be inefficient and ineffective.

Fang Zhou Hai Pham Jianhui Yue Hao ZouWeikuanYu[3]: Hadoop used to have limitations in processing and storing file sizes, particularly when it came to small files. To address this issue, SF MapReduce offers two methods: Small File Layout and Specialized MapReduce. The former is utilized by HDFS to improve storage and I/O performance. Additionally, the SF MapReduce framework expands the capabilities of MapReduce beyond large Hadoop files, allowing for increased throughput and reduced memory usage on the Hadoop Name Node. In comparison to the original Hadoop and HAR configurations, SF MapReduce enhances MapReduce processing by 14.5x and 20.8x, respectively.

DrAnjaliMathur[4]: The purpose of "HDFS" is to store and access massive amounts of data while providing high fault tolerance. It utilizes a cloud-based storage foundation that offers scalable capacity, optimal performance efficiency, and affordable storage costs. The support for parallel processing results in high throughput, making it an ideal choice for applications that involve large datasets. However, using HDFS for storing numerous small files can lead to processing issues and inefficiencies. The research work included a comparison table that evaluates the performance throughput of various existing approaches.

Liang Huang, Jun Liu, Weixiu Meng [5]: The electronic information industry is expanding rapidly and the amount of data being produced is increasing at an alarming rate due to the surge in internet traffic. The traditional storage systems are not equipped to handle the immense volume of data being generated. Consequently, Hadoop has emerged as a widely-used big data platform that offers robust data storage and distributed computing capabilities. In addition, newly developed software frameworks for distributed processing are known for their ability to manage and store vast quantities of data.

Wenjun Tao; YanlongZhai; Jude Tchaye-Kondi [6]:HDFS is highly scalable and reliable for managing large files across numerous servers, but it experiences significant slowdowns when handling many small files. Although there have been efforts to address this problem, the current solutions, such as HARs, sequence files, and map files, are not effective in reducing name node memory usage and improving access efficiency. However, one approach that HDFS uses to manage large numbers of small files is by aggregating them into larger files and building scalable indexes based on linear hashes. This approach offers several benefits, including reducing metadata size, eliminating the need for client-side file reordering, enabling the insertion of additional small files into the merged file, and delivering excellent access performance.

ChandrasekarS,DakshinamurthyR,SeshakumarPG,PrabavathyB,ChitraBabu[7]:A technique called Extended Hadoop Distributed File System (EHDFS), which draws on research by Dong et al., was utilized to enhance the effectiveness of storing and retrieving small files in HDFS. EHDFS consolidates a group of linked files identified by the client into a single large file, thereby reducing the number of files. Additionally, an indexing strategy was developed to facilitate access to different files from joined files. To improve I/O efficiency and alleviate the name node, index prefetching is also available. According to test results, EHDFS can efficiently store and access numerous small files while decreasing the Name Node's primary memory metadata footprint by 16%.

Tharwat,MohammedBadawy,andAyman[8]: Various approaches have been developed to address the issue of poor performance in handling and accessing small files in Hadoop. These methods include HBase, S3DistCp, Federated Name Nodes, batch file integration, sequence files, and Hadoop archive files. Despite their respective benefits, they also share certain drawbacks. For instance, the duration of converting all HDFS files to sequence files may not be proportional to the amount of time required to access files stored in sequence files. To address this limitation, the Small Files Search and Aggregation Node has been introduced as an extension of the Sequence Files Technique, which is elaborated in this document.

NeetaAlangeandAnjaliMathur[9]: The amount of data being generated is increasing exponentially as more of the world conducts business online. These vast quantities of data are challenging to process and analyse quickly. To store, process, and generate these large volumes of data, the industry is increasingly relying on Hadoop frameworks that distribute the data across servers. However, handling these large amounts of data, especially when it comes to small file sizes and storage optimization, remains a significant challenge.

KyoungsooBok,HyunkyoOh,JongtaeLim,JaesooYoo [10]: Over the past few years, analysing limited quantities of data has become essential to offer personalized updates and improved services to individual users. To address this need, we suggest using a distributed cache management strategy that takes into account the cache metadata, for efficient access to Tiny Files (HDFS) from the Hadoop Distributed File System. Our proposal involves storing multiple small files in a single chunk, which minimizes the number of metadata that the name node needs to manage. Additionally, we reduce wasteful access by maintaining client and data node caches for requested files and synchronizing information in client caches through communication cycles.

B.santhoshkumar,P.kanagaranjitham,K.r.karthekk,J.gokila[11]: To enhance Hadoop's efficiency when dealing with small files, several strategies have been proposed in various research articles. However, these approaches have significant limitations. As a result, alternative and effective methods like the SIFM and Merge models have emerged as the preferred ways to handle small files in Hadoop. Additionally, the recently introduced HAR is another crucial technique for optimizing Hadoop's performance.

Liu Changtong[12]: Hadoop enables open source distributed computing, while its HDFS component provides ample storage capacity that makes it well-suited for cloud storage systems. However, HDFS was initially designed for handling large software pieces through streaming access, leading to lower storage efficiency for numerous smaller files. To address this challenge, HDFS storage processes have been improved. Before uploading to the HDFS cluster, files are evaluated, and if they are smaller, they are merged, with index information stored as key-value pairs in the index file. As a result, simulations demonstrate that the updated HDFS consumes less name node RAM (HAR files) than the original HDFS and Hadoop archives.

Ahad,Mohd&Biswas,Ranjit[13]: The voluminous generation of data resulting from recent technological advancements in computing cannot be effectively handled by traditional tools, processes, and systems. Consequently, new techniques and frameworks have emerged to manage big data, among which Hadoop is a prominent framework. Hadoop offers efficient storage, retrieval, processing, and analytics of large files. However, when processing hundreds or thousands of small size files, Hadoop's performance tends to degrade. This paper examines the challenges and opportunities associated with managing a large number of small size files and reviews various techniques available for efficiently handling small files in Hadoop based on performance parameters such as access time, read/write complexity, scalability, and processing speed.

LianXiong,YuanchangZhong,Xiaojun Liu, AndLiu Yang[14]: Due to the rapid progress of smart health, the use of wearable technology and health sensors results in the generation of a substantial amount of spatio-temporal data files, which are scattered across multiple servers and negatively impact system I/O performance. Various methods exist for resolving the issue of small files, but most are designed for particular applications. In the case of utilizing sensor data for smart health, these methods prove ineffective due to the nature of user access behaviour and data types. To address this challenge, this approach evaluates the properties of health sensor data and user access preferences, and employs spatio-temporal clustering on historical user access data.

Awais Mehmood, Waqas Mehmood [15]: This study investigates how HDFS manages the storage and access requirements of numerous small files by analysing its architecture, mechanics, and behaviour. To address this issue, Hadoop can consolidate all small files into a single HAR file and treat it as a large file. This white paper extensively delves into the functioning and significance of HAR, along with its cost-benefit analysis.

YonghuaHuo,ZhihaoWang,XiaoXiaoZeng,YangYang,WenjingLi,ZHONGCheng[16]: HDFS is intended for the storage of large files, but its performance declines when handling numerous small files. To tackle this issue, various strategies have been developed. This study proposes using a Small File Server (SFS) as additional hardware between the user and HDFS to address the small file problem. The approach involves utilizing a file merging technique that relies on temporal continuity, an index structure for small file retrieval, and a prefetching mechanism to enhance file read and write speed.

KunWang,YangYang,XuesongQiu,ZhipengGao[17]: Hadoop has become a popular big data processing platform due to its stability and scalability. Although HDFS is capable of storing large files efficiently, the "small file problem" arises when numerous small files are stored due to the limited storage and access performance of the Name Node. To address this issue, multi-layer optimized memory technology (MOSM) has been developed to improve small data storage and eliminate bugs associated with storing large numbers of small files. An algorithm that combines smaller files into larger ones is used to reduce the memory consumption of the Name Node. Once merged, the prefetch cache mechanism and effective hybrid indexing scheme speed up access to small files.

R.Rathidevi,R.Parameswari[18]: The rapid progress in technology leads to the creation of numerous files daily from various sources. The accumulation of these files takes up significant disk space for storage. Besides large files, a considerable number of small files are also considered big data. For processing large files, Hadoop is a popular tool. However, processing small files using Hadoop can be challenging because it reserves 128MB of storage space for each record. To tackle this problem, the CSFC (centroid-based clustering of small files) approach is used, which groups small files together for more efficient processing.

Priyanka Phakade, Dr.SuhasRaut[19]: As the use of the Internet continues to grow, more individuals are seeking to store their data on cloud computing platforms. However, because user data is often comprised of small files, HDFS (Hadoop Distributed File System) struggles to manage large numbers of them. The current system's mapping tasks process only one block of input at a time, resulting in each mapping task receiving less input if there are numerous small files. As a result, HDFS's capacity to handle many small files is compromised. Nonetheless, small files can be effectively managed by HDFS in the proposed system. When a client requests for a small file to be stored in HDFS, the Name Node may combine the files into one split and use it as input for the mapping job. Some reducers may take intermediate outputs from mapping operations as input, with the resulting combined output being reordered by the Reducer. This approach results in reduced processing time due to fewer mapping jobs.

ChatupornVorapongkitipun,NatawutNupairoj[20]: The HDFS architecture depends on a single master, namely the Name Node, to manage information for many slaves, or Data Nodes. However, due to this reliance, the Name Node may become a bottleneck, particularly when dealing with numerous small files. To ensure efficiency, the Name Node typically stores all HDFS metadata in main memory. Nonetheless, an abundance of small files may exceed the name node's capacity. To address this issue, a mechanism called New Hadoop Archive (NHAR) is proposed in this study, which utilizes a Hadoop Archive (HAR) to optimize access efficiency for small files in HDFS and enhance metadata memory consumption. Moreover, HAR functionality is extended to enable new files to be added to existing archive files. Our experimental findings demonstrate that our approach notably improves small file access efficiency, surpassing HAR by up to 85.47%.

RaveenaAggarwal ,JyotiVerma,Manvi Siwach[21]: The Apache Hadoop open source software library incorporates a range of software tools to handle the storage and processing of small files. A small file is defined as a file that is considerably smaller than the HDFS standard block size, and because each small file necessitates a separate block, this can result in increased storage space usage and longer access and processing times. There comes a point beyond which it is not feasible to scale storage, tolerate access latency, or deal with processing latency.

Bende,Sachin&Shedge,Rajashree[22]: Hadoop has experienced a significant surge in usage in recent years, and its adoption is widespread. Notable big users, such as Yahoo, Facebook, Netflix, and Amazon, rely on Hadoop primarily for unstructured data analysis. The Hadoop framework is well-suited for handling both structured and unstructured data. Although Hadoop's distributed file system (HDFS) is designed for storing large files, it encounters difficulties when storing a large number of small files, as all files are managed by a single server. Several approaches have been suggested to address the issue of small file management in HDFS. This article presents a comparative analysis of various methods for resolving the small file problem in HDFS.

TanviGupta,Prof.SSHandas[23]: The HDFS has a limitation on the number of files it can hold, which is determined by the size of the main memory of the Name Node. Moreover, when there is no prefetching strategy in place, HDFS disregards associations between files, leading to poor I/O performance for user accounts. To address the issue of storing and accessing small files in HDFS efficiently, the Extended Hadoop Distributed File System (EHDFS) was developed, building on the work of Dong et al. EHDFS reduces the number of files by consolidating linked files into a large file. Chandrasekar S. and colleagues developed an indexing system to enable easy access to component files from their corresponding combined files. Index prefetching is also provided to enhance I/O rates and minimize the burden on name nodes. Additionally, "Avatar Nodes" by S. Chandra Muriswaran et al. are utilized to eliminate single points of failure and increase storage capacity. The purpose of this document is to enhance efficiency by introducing the concept of "avatar nodes" as an indexing method for managing small files in HDFS in the event of primary name node failure.

YanlongZhai,JudeTchaye-Kondi,Kwei-JayLin,LiehuangZhu,WenjunTaoXiaojiangDu,MohsenGuizani[24]: To extract data from smaller files within a larger archive file, an index file is employed to store relevant information. Nevertheless, current archive-based solutions suffer from substantial overhead when retrieving file content. This is due to the need to retrieve retrieval metadata prior to accessing the file's contents, resulting in increased processing and I/O and decreased efficiency. However, by obtaining metadata directly from the corresponding section of the index file, HPF minimizes access overhead. This results in the requirement of fewer additional processing and I/O resources, ultimately leading to faster archive file access. Our indexing system employs two hashing algorithms.

BoDong,QinghuaZheng,FengTian,Kuo-Ming Chao,RuiMa,RachidAnane[25]: Hadoop Distributed File System is a commonly used tool in supporting internet services. However, native HDFS struggles when working with small batches of files, despite its popularity. This white paper investigates the root cause of this issue, which stems from the heavy load that a large number of small files places on the HDFS name node. The paper suggests that HDFS does not consider correlations between small files when placing data, and offers mechanisms such as prefetching to optimize I/O speed. Furthermore, the paper explores the definition of "small" files in the context of HDFS and conducts experiments to determine the appropriate size classification. Ultimately, the paper classifies files into three groups based on their correlation characteristics: structurally related files, logically related files, and independent files.

III. Conclusion:

This document presents the idea of merging small files to decrease their memoryusage, along with the use of Hadoop archive technology and sequential file access. Various techniques are employed to enhance map files, optimize access methods, and improve the data stored in Hadoop clusters. However, accessing files in Hadoop still requires further improvements. In the future, advancements in archive file technology are expected to enhance data access and storage efficiency.

Detecting file sizes prior to merging different files can enhance the efficiency of the merge process and memory allocation. An upcoming improvement of this concept involves utilizing this technique for accessing small image files.

Diagram:

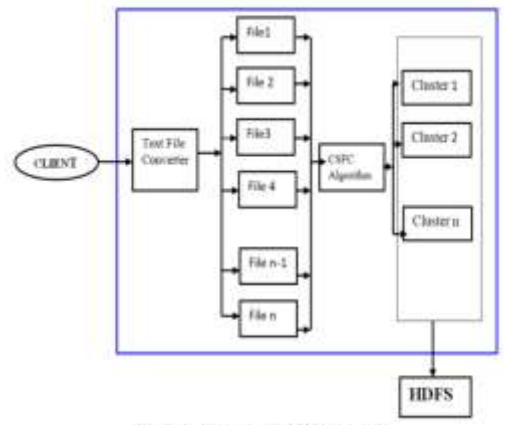

Fig 2. Architecture of CSFC Approach

References:

[1]. **LiuJiang,BingLiandMeinaSong,**"Theoptimization of HDFS based on small files," 20103rd IEEE International Conference on BroadbandNetworkandMultimediaTechnology(IC-BNMT),2010, pp. 912-915,doi:10.1109/ICBNMT.2010.5705223.

[2]. **XianzhenRen,XiuhuaGeng,YiZhu.**"AnAlgorithmofMergingSmallFilesinHDFS,"20192ndInternationalConferenceonArtificialIntelligenceandBigData(ICAIBD),2019,pp.24-27,doi:10.1109/ICAIBD.2019.8836971.

[3]. **Fang Zhou, Hai Pham, Jianhui Yue, Hao Zouand Weikuan Yu,** "SFMapReduce: An optimizedMapReduceframeworkforSmallFiles,"2015IEEEInternationalConferenceonNetworking,Architecture and Storage (NAS), 2015, pp. 23-32,doi:10.1109/NAS.2015.7255218.

[4]. **N.AlangeandA.Mathur**,"SmallSizedFileStorageProblemsinHadoopDistributedFileSystem," 2019 International Conference on SmartSystems and Inventive Technology (ICSSIT), 2019,pp.1202-1206,doi:10.1109/ICSSIT46314.2019.8987739.

[5]. **L. Huang, J. Liu and W. Meng,** "A Review ofVariousOptimizationSchemesofSmallFilesStorage on Hadoop," 2018 37th Chinese ControlConference(CCC),2018,pp.4500-4506,doi:10.23919/ChiCC.2018.8483588.

[6]. **W. Tao, Y. Zhai and J. Tchaye-Kondi**, "LHF: ANewArchiveBasedApproachtoAccelerateMassiveSmallFilesAccessPerformanceinHDFS,"2019 IEEE Fifth International Conference on BigDataComputingServiceandApplications(BigDataService),2019,pp.40-48,doi:10.1109/BigDataService.2019.00012.

[7]. **S.Chandrasekar,R.Dakshinamurthy,P.G.Seshakumar,B.PrabavathyandC.Babu**,"Anovelindexingschemeforefficienthandlingofsmallfilesin HadoopDistributedFileSystem,"2013InternationalConferenceonComputerCommunicationandInformatics,2013,pp.1-8,doi:10.1109/ICCCI.2013.6466147.

[8]. **T.El-Sayed,M.BadawyandA.El-Sayed**,"SFSAN Approach for Solving the Problem of SmallFilesinHadoop,"201813thInternationalConferenceonComputerEngineeringandSystems(ICCES),2018,pp.135-138,doi:10.1109/ICCES.2018.8639479

[9]. **N. Alange and A. Mathur**, "Small Sized FileStorageProblems in HadoopDistributed FileSystem," 2019 International Conference on Smart SystemsandInventiveTechnology(ICSSIT),2019,pp.1202-1206,doi:10.1109/ICSSIT46314.2019.8987739.

[10]. **KyoungsooBok,JongtaeLim,HyunkyoOhand JaesooYoo**, "An efficient cache managementschemeforaccessingsmallfilesinDistributedFileSystems," 2017 IEEE International Conference onBig Data and Smart Computing (BigComp), 2017,pp. 151-155, doi:10.1109/BIGCOMP.2017.7881731.

[11]. **.B. S. Kumar, P. K. Ranjitham, K. R. KarthekkandJ.Gokila**,"Surveyonvarioussmallfilehandlingstrategieson Hadoop,"2016InternationalConferenceonCommunicationandElectronicsSystems(ICCES), 2016, pp.1-4, doi:10.1109/CESYS.2016.7889882.

[12]. **L. Changtong,** "An improved HDFS for small file," 2016 18th International Conference on Advanced Communication Technology (ICACT), PyeongChang, Korea (South), 2016, pp. 474-477, doi: 10.1109/ICACT.2016.7423438.

[13]. **Ahad, Mohd & Biswas, Ranjit.** (2019). Handling Small Size Files in Hadoop: Challenges, Opportunities, and Review. 10.1007/978-981-13- 0514-6_62.

[14]. **Xiong,Lian&Zhong,Yuanchang&Liu,Xiaojun&Yang,Liu**.(2019).ASmallFileMergingStrategyforSpatiotemporalDatainSmartHealth.IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2893882.

[15]. **Mehmood,M.Usman,W.MehmoodandY. Khaliq**, "Performance efficiency in Hadoop forstoring and accessing small files," 2017 SeventhInternationalConferenceonInnovativeComputing Technology (INTECH), 2017, pp. 211-216,doi:10.1109/INTECH.2017.8102449.

[16]. **Y.Huo,Z.Wang,X.Zeng,Y.Yang,W.LiandZ. Cheng**, "SFS: A massive small file processingmiddleware in Hadoop," 2016 18th Asia-PacificNetworkOperationsand ManagementSymposium(APNOMS),2016,pp.1-4,doi:10.1109/APNOMS.2016.7737234.

[17]. **K. Wang, Y. Yang, X. Qiu and Z. Gao**, "MOSM:An approach for efficient storing massive smallfilesonHadoop,"2017IEEE2ndInternationalConference on Big Data Analysis (ICBDA), 2017,pp.397-401,doi: 10.1109/ICBDA.2017.8078848.

[18]. **Rathidevi,R.&Parameswari,R**(2020).Performance Analysis of Small Files in HDFS usingClusteringSmallFilesbasedonCentroidAlgorithm. 640-643. 10.1109/I-SMAC49090.2020.9243418.

[19]. **Priyanka Phakade, Dr.SuhasRaut**, " AnInnovative Strategy for Improved Processing ofSmall Files in Hadoop " , International Journal ofApplicationorInnovationinEngineering&Management(IJAIEM),Volume3,Issue7,July2014 ,pp.278-280,ISSN2319–4847

[20]. **C.VorapongkitipunandN.Nupairoj**,"Improving performance of small-file accessing inHadoop," 201411thInternationalJointConferenceonComputerScienceandSoftwareEngineering(JCSSE),2014,pp. 200-205,doi:10.1109/JCSSE.2014.6841867.

[21]. **RaveenaAggarwal,JyotiVerma,ManviSiwach**,Smallfiles'probleminHadoop:Asystematic literature review,Journal of King SaudUniversity-ComputerandInformationSciences,Volume34,Issue10,PartA, 2022,Pages 8658-8674,ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2021.09.007. (https://www.sciencedirect.com/science/article/pii/S1319157821002585).

[22]. **Bende,Sachin&Shedge,Rajashree**.(2016).DealingwithSmallFilesProbleminHadoopDistributedFileSystem.ProcediaComputerScience. 79. 1001-1012. 10.1016/j.procs.2016.03.127.

[23]. **T. Gupta and S. S. Handa**, "An extended HDFSwithanAVATARNODEtohandlebothsmallfilesandtoeliminatesinglepointoffailure,"2015InternationalConferenceonSoftComputingTechniques and Implementations (ICSCTI), 2015,pp.67-71,doi:10.1109/ICSCTI.2015.7489606.

[24]. **Zhai, Yanlong&Tchaye-Kondi, Jude & Lin,Kwei-Jay&Zhu,Liehuang&Tao,Wenjun&Du,Xiaojiang&Guizani, Mohsen**. (2021). HadoopPerfectFile:Afastandmemory-efficientmetadataaccess archive file to face small files problem inHDFS.JournalofParallelandDistributedComputing.156.10.1016/j.jpdc.2021.05.011.

[25]. **BoDong,QinghuaZheng,FengTian,Kuo-MingChao,RuiMa,RachidAnane**,Anoptimized https://doi.org/10.1016/j.jnca.2012.07.009.(https://www.sciencedirect.com/science/article/pii/S1084804512001610)