International Journal for Housing Science and Its Applications
Publish August 6, 2025. Volume 46, Issue 3 Pages 5338-5345

https://doi.org/10.70517/ijhsa463460

Research on optimization scheme for large-scale digital media data processing based on distributed computing architecture

Tao Liu[1,*] and Meiling Yang[2]
[1] Department of Journalism and Communication, Anhui Vocational College of Press and Publishing, Hefei, Anhui, 230601, China
[2] Hefei Transportation Comprehensive Administrative Law Enforcement Detachment, Hefei Transportation Bureau, Hefei, Anhui, 230601, China
Corresponding authors: (e-mail: liutao5268@163.com).

Abstract The explosive growth of digital media data makes the traditional centralized processing architecture face serious challenges in computational efficiency and storage cost. This paper responds to the demand for efficient processing of large-scale digital media data and launches a research and analysis based on distributed computing architecture. Combing the feature clustering process of distribution sets and features of association rule data, as well as the distributed data frequent item clustering collection process. At the same time, the CDMDP protocol with advanced encryption technology, distributed storage mechanism and smart contract features is designed to effectively realize the distribution and protection of digital data content. Combining logical search tree and parallel algorithm SFUPM-SP, Spark-based parallel mining algorithm for distributed computing of big data is proposed as a processing and optimization method for large-scale digital media data. In the system platform built by this paper's method, the average execution time of K-Means algorithm on data is only 17.9 seconds, which demonstrates the effectiveness and feasibility of this paper's method.

Index Terms distributed computing architecture, CDMDP protocol, data processing, parallel mining algorithm, K-Means algorithm

I. Introduction

In recent years, with the wide application and development of information technology, the role of digital media has become increasingly significant in contemporary society. Digital media not only includes traditional text, image and audio content, but also covers a variety of forms such as video, interactive content, virtual reality (VR) and augmented reality (AR) [1]. However, digital media are often characterized by high security requirements, high processing complexity and huge data volume when processing and distributing [2]. Cloud computing technology, as an emerging information technology, provides new ideas for solving these problems by virtue of its high scalability and flexibility as well as powerful computing power [3], [4].

Cloud computing technology is essentially a hybrid technology formed during the development of the information age, involving content based on distributed computing, parallel computing, data storage, etc., and can be decomposed into multiple small programs using the Internet platform to successfully complete the processing of data [5]-[8]. In the computer data processing work, cloud computing technology can synchronize the screening of different data information and classify it according to the corresponding requirements, so that the query work also becomes more efficient [9]-[11]. Also, because in the cloud computing processing, user information does not require continuous maintenance of technology and related software, the processing cost of digital media information is significantly reduced [12], [13]. At the same time, with the help of cloud computing technology, customers can store data and information on the cloud, and the introduction of firewalls, antivirus software and other security means to carry out security prevention and control work, which significantly improves the integrity and security of the amount of data in the processing [14]-[17]. Therefore, cloud computing technology is introduced into digital media processing and distribution to analyze its positive role in practical application, such as optimizing resource allocation, enhancing data security, etc., so as to provide theoretical support and practical guidance for the application and promotion of cloud computing technology in the field of digital media [18]-[20].

This paper firstly discusses in detail the distribution set of association rule distributed data and the feature clustering process, and further analyzes the clustering collection process of distributed data in frequent projects. Secondly, it describes the operation process of improving CDMDP protocol as a secure distribution and protection method for digital content data. Propose again the parallel mining algorithm for distributed computing of big data based on Spark, describe the content of logical search tree and the mathematical principle of parallel algorithm to make up for the defects of existing big data mining algorithms with poor computational model. Finally, the effectiveness of the proposed digital media data processing method is examined through system platform performance tests, different scenario tests and program performance comparison experiments, respectively.

II. Digital Media Data Processing Based on Distributed Computing

II. A. Association rule distribution set and feature clustering process

The association rule distribution set and feature clustering process mainly includes:

(1) Association rule distribution set process, the task is to establish the association rule distribution set of distributed data by mining and analyzing the semantic features of distributed data by using the technical method of fuzzy data feature detection, combined with the investigator's preference selection for data intelligence investigation. The association rule distribution set of distributed data can be expressed by functional equation (1):

$$W = \frac{\sum_{i=1}^{s} W_i \frac{1}{d_i^k}}{\sum_{i=1}^{s} \frac{1}{d_i^k}} \tag{1}$$

where, W_i denotes different distributed data that have been processed by the technology, d_i^k denotes the investigator's preference choice for data intelligence investigation, s denotes the frequent item association clustering set of distributed data, and k denotes the similarity coefficient of data nodes between different distributed data.

(2) Association rule feature clustering process, the task is to extract the amount of multiple data association features of different distributed data by mining, take the semantic data features between different distributed data as the association articulation point, and perform association clustering on their association rule features.

II. B. Distributed Data Frequent Project Clustering Aggregation Process

The task of distributed data frequent project clustering collection process is to realize effective fusion of various distributed algorithms with the preset value requirements of different data intelligence investigations, mainly based on the original data association feature quantity between different distributed data, and construct different clustering collections of distributed data with the same or similar neighboring coefficients, so as to improve the high efficiency of distributed algorithmic data intelligence investigations. The investigators can utilize the data clustering recursive function formula (2)-(3):

$$W_i(t) = W_i(t-1) - \mu^t \times D\left(W_i\left(t-1\right), X_{nt}\right) \tag{2}$$

$$i = \arg\min d\left(W_i(t-1), X_{nt}\right) \tag{3}$$

Operational mining of distributed data frequent item clustering sets, where $W_i(t-1)$ denotes the clustering center of distributed data, X_{nl} denotes the test sample set of distributed data, $D\left(W_i(t-1), X_{nt}\right)$ denotes the clustering distance between the clustering center of the pre-determined data and the tested distributed data sample set, and μ^l denotes the convergence speed of different distributed data being clustered. Obviously, the essence of the distributed data frequent project clustering collection process construction is to mine to obtain the potential correlation structure or mathematical relationship between different distributed data, and then the data adjacent to the similarity coefficients of the same or similar to the clustering division, thus preventing the data turbulence or data fitting and other phenomena in the process of distributed computing.

II. C. CDMDP protocol improvements

Cloud computing technology has greatly enhanced the capability of digital media processing and distribution, however, ensuring the copyright security of digital media content in cloud platforms has been a key challenge in this area. The existing Digital Media Distribution Protocol (DMDP) aims to ensure the integrity and security of digital content in the distribution process, but its efficiency and security need to be further improved in the cloud computing environment. For this reason, this paper optimizes DMDP, proposes CDMDP protocol adapted to cloud computing architecture, and introduces the SPPetri model to verify its security, aiming to break the restriction of copyright protection issues on digital media network distribution.

The CDMDP protocol, i.e., the DMDP protocol in cloud computing environment, is improved in several aspects relative to the DMDP protocol. First, the CDMDP protocol pays more attention to the characteristics of the cloud computing environment, such as multi-tenancy characteristics and dynamics of resources, to ensure the efficient distribution of digital content in the cloud computing environment. Secondly, the CDMDP protocol increases the strength of copyright protection for digital media, by introducing more advanced encryption and authentication means to ensure the security of digital content in the process of distribution. The details of the CDMDP protocol flow are shown in Fig. 1.

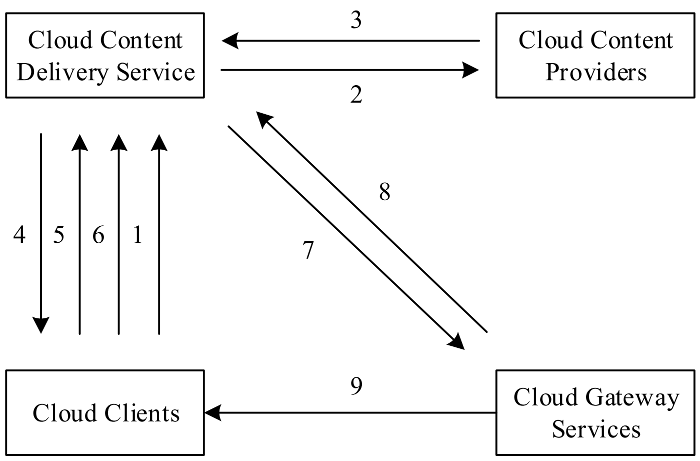

Figure 1: CDMDP protocol process

Steps 1, 2, and 3 are for the cloud computing client to submit the required content information and related claim information, and the content provider integrates this information into a composite digital object and sends it to the content distributor. Next, steps 4 and 5 are for the content distributor to encrypt this composite digital object and return the encrypted object to the cloud computing client. Steps 6 and 7 are for the client to submit the account information of the content provider and the user to the cloud computing charging gateway after receiving the encrypted composite digital object. Steps 8 and 9 are for the cloud computing platform to carry out the corresponding charging process according to the submitted account information, thus complete the whole transaction process.

In terms of security, the CDMDP protocol employs a variety of encryption and authentication means, such as public and private key encryption, hash function integrity checking, and digital signature technology, to ensure the security of the protocol. These improvements make the distribution of digital media in the cloud computing environment more secure and efficient, and the CDMDP protocol not only effectively protects the copyright of digital media, but also significantly improves the distribution efficiency and reduces the distribution cost. In addition, the protocol has good scalability and compatibility, and can be well adapted to different sizes of cloud computing environments and different digital media distribution needs.

II. D.Spark-based Parallel Mining Algorithm for Big Data Distributed Computing

The Spark-based parallel mining (SFUPM-SP) algorithm is proposed to address the problems such as poor Ma-pReduce distributed computation model that exists in big data analysis mining algorithms for databases. The algorithm efficiently traverses all itemsets by constructing a logical search tree structure to avoid duplicating or omitting the computed itemsets in order to implement the pruning strategy.

II. D. 1) Logical search tree

In general, traditional stand-alone algorithms mostly rely on data structures such as lists and projections to traverse the itemsets during the mining process. However, these traditional data structures show their inherent limitations in some cases, as follows.

(1) Memory limitation: traditional list and projection structures require complete loading of data into memory, which may lead to memory overflow and affect the system performance of list and projection structures.

(2) Performance efficiency limitations: the efficiency of the list may be affected when performing frequent element insertion or deletion operations, mainly due to the need to adjust the order of the internal elements to maintain the consistency of the data structure, resulting in an increase in the time complexity of the operation.

(3) Limitations of Distributed Processing: distributed systems become critical in large-scale data processing. Distributed systems allow data sets to be partitioned into small chunks and processed in parallel on multiple computing nodes.

II. D. 2) Parallel algorithm SFUPM-SP design

The SFUPM-SP algorithm is a distributed running algorithm implemented using the Spark framework. In the initial phase of the algorithm, the original dataset is divided into data blocks and stored in HDFS. Subsequently, these data chunks are transformed into initial RDDs through the Application Programming Interface (API) provided by SparkCore.The algorithm then processes the data using two operators, transformation and action, and simultaneously performs the transformation operations of the RDDs to meet specific computational requirements.

(1) Input part: the SFUPM-SP algorithm requires as input a public transactional database D stored in a distributed file system, HDFS, and a corresponding income statement.

(2) Preprocessing phase: in this phase, the dataset is first read from the HDFS path using the textFile method provided by the SparkContext object and stored in a partitioned manner to create an initial HDFS-RDD. next, each row is operated on by the flatMap operator provided by Spark, extracting the name of each item, the transactional utility and the utility that constructs a ListBuffer containing the ternary. Subsequently, elements with the same key are generalized using the reduceByKey operator, and the sum, quantity, and total number of the same key are obtained through the accumulation operation. Then, the result of the generalization is mapped into a new RDD using the map operator, containing three items: itemset, frequency and transaction weighted utility.

(3) Mining phase: in this phase, the algorithm has to compute the utility, frequency, subtree utility and local utility of each itemset and output the result into HDFS. This phase covers the use of two pruning strategies. The whole process is performed selectively until no more candidate sets are generated and the iteration is terminated. In each iteration, the itemsets are extracted from the iteration file and judged using arrays, and the eligible itemsets are placed into a pool of potential Skyline frequent-utility patterns. Eventually, frequent-utility patterns are generated from the pool.

III. Application and testing of digital media data processing methods

III. A. System performance testing

The performance test of the system is mainly by comparing with Mahout of Hadoop cluster, using the same training dataset for the same algorithm model training and comparing the execution time. The test selects K-means algorithm, based on different number of computing nodes and different computing platforms to perform clustering operation on the same data set. After statistics, the experimental test results are shown in Figure 2.

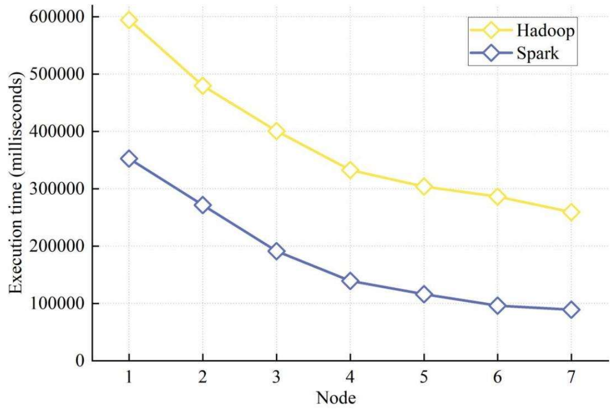

Figure 2: Performance comparison under hadoop and spark

As can be seen from Fig. 2, the performance of the distributed computing based digital media data processing method designed in this paper is superior to Hadoop.In this experiment, the average execution time of K-means algorithm is 17.9 seconds on this system, while the average execution time on the native Hadoop cluster is 37.9 seconds, which shows that the memory-based iterative computation is very fast and saves a lot of disk operations, the performance improvement is extremely obvious. At the same time, the user can configure the workflow and set the algorithm parameters through the web interface, which reduces the difficulty of machine learning applications and development time. Test results show that the method in this paper meets the performance requirements.

III. B. Complex multi-type modification scenario testing

III. B. 1) Test analysis of incremental processing

Before conducting experiments, this paper will first analyze the various aspects of incremental processing. An incremental computing task mainly has five phases such as submit, Map, Shuffle, Reduce, and output, etc.

Currently, the core of most incremental computing frameworks are in the Map phase, which improves the execution efficiency of the job by adding some auxiliary operations in the phase, while more time-consuming time is needed in the submit and output phases to perform preprocessing and post-processing to support the work of incremental processing. In this paper, we define the total efficiency improvement as E_t as in equation (4):

$$E_t = \frac{T_2 - T_1 - T_3}{T_{\mathrm{mr}}} \tag{4}$$

where T_{mr} denotes the elapsed time of a normal MapReduce execution of a job. T_1 refers to the additional elapsed time for Spark preprocessing. T_2 refers to the time reduction by Spark in the Map phase. T_3 refers to the additional elapsed time of Spark post-processing, which mainly removes useless sliced results in preparation for the next computation.

To summarize, the size of the input dataset is linearly and positively correlated with T_1 in Fig. 3, which depicts how T_1 varies as the input dataset progressively grows. As can be seen from the figure, the trend depicted by the rest of the points is very clean, except for the few points towards the front which are more unstable. It is obvious that the slope posed by T_1 and the size of the dataset is very flat compared to the size of the growth of the dataset. This means that the time-consumption of the preprocessing phase does not put a huge burden on the system time-consumption due to the growing dataset. This elapsed time generated by preprocessing is predictable in that it steadily maintains a low level of growth as the dataset grows.

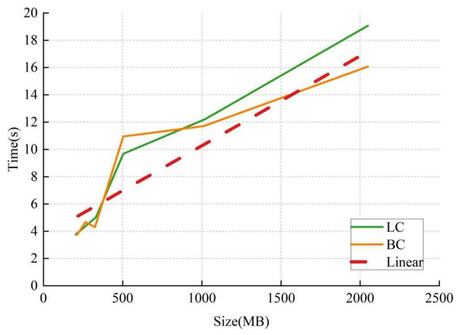

Figure 3: The time-consuming preprocessing under different datasets

Figure 4 depicts the trend of T_3 as the dataset grows. In post-processing, Spark will clean up useless data in the final stage of the job run. For example, in the result cache, if a record is not accessed in this run of the device, it means that the record has been modified and will not be accessed in the future, so Spark should delete this record and delete the Task result that this record points to as well. When the randomness of data modification does not vary much, T_3 is linearly and positively correlated with the number of modification operations. This is because the more modification operations there are, the more slices will be modified and the system will not be able to retrieve their records in the result cache, so that the computation result files pointed to by these records will have to be deleted altogether.

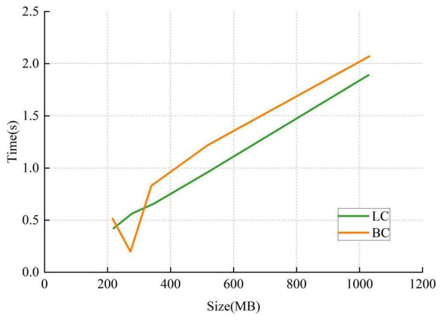

Figure 4: The time consumption of post-processing under different datasets

III. B. 2) Performance testing in different scenarios

In order to verify that the method (Spark) in this paper has high computational efficiency and excellent stability, this section will go deeper into the Task level and select ChkReuse and PlainMR as the comparison methods to observe the time consumption of each method. First, the dataset is randomly modified so that the update rate of the dataset grows from 0.5% to 7%. Then the elapsed time of the task is recorded and the variation curve of elapsed time is plotted in Fig. 5 for different update rates.

At any update rate, PlainMR needs to be recomputed, so the elapsed time is essentially constant. As the update rate grows, the elapsed time of ChkReuse becomes very unstable, this is because ChkReuse is based on fixed-length division, the larger the update rate is, the larger the difference between the old division and the new division is, and the more chunks of data blocks need to be recalculated, and when the update rate grows to a certain height, the efficiency of fine-grained reuse is even worse than that of coarse-grained reuse. Since Spark is based on content division, its division is more stable even if the update rate grows. When the update rate is less than 4%, Spark significantly outperforms ChkReuse and PlainMR in terms of task time (<1.5s) and stability.

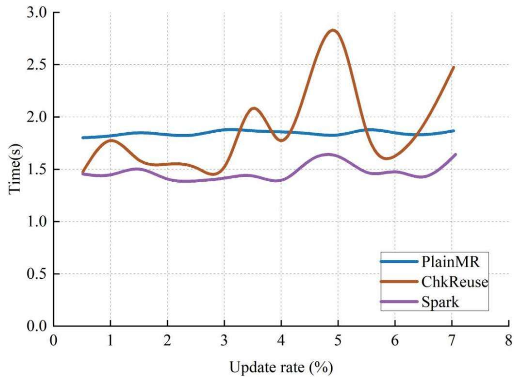

Figure 5: The map task takes time under different updates

The number of chunks that need to be recalculated under different update rates is shown in Figure 6. The number of chunks that need to be recalculated under different update rates for ChkResue is significantly more than that for Spark. From the stability point of view, as the update rate grows, ChkResue oscillates drastically and rises, while Spark rises steadily and slowly and the maximum number of chunks does not exceed 100, which is undoubtedly better than the stability of Spark. Spark is undoubtedly more stable.

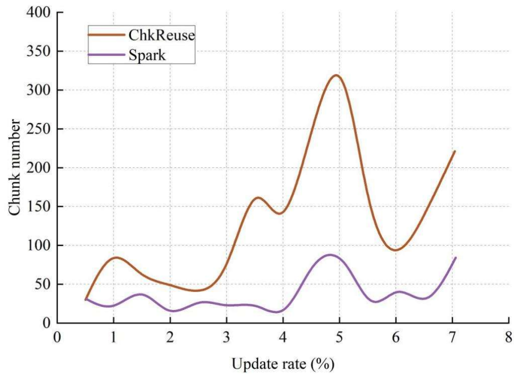

Figure 6: The number of chunks to be recalculated under different update rates

Also, it should be noted that the test data is randomly modified on the basis of metadata, and Figures 5 and 6 are violently oscillated at an update rate of 5%, which is due to the scattered distribution of the updates, resulting in more chunks changing, which leads to a sudden increase in the amount of computation.

III. C. *Program performance comparison experiment*

III. C. 1) Comparison of Recognition Classification Performance between Mapper and Reducer Side

Test dataset X1 on the application cluster. The data size of dataset X1 is 50G, and the Block Size is 128M in HDFS storage, the appropriate SplitSize can be configured by setting the appropriate min/maxsplitSzie in

MapperReducer as in Equation (5), so as to complete the comparison of the performance of the recognition and classification on the Mapper/Reducer side. SplitSize customization settings.

$$SplitSize = \max\left\{\min split \quad Size, \min\left\{\max split \quad Size, Block \quad Size\right\}\right\} \tag{5}$$

Table 1 shows the results of comparing the performance of Mapper and Reducer side recognition and classification.

Table 1: Comparison of recognition and classification performance at different ends

	SplitSize: Block size	Execution time
Mapper	64M (default)	33m07s
Mapper	128M	24m28s
Mapper	256M	21m07s
Mapper	512M	21m24s
Reducer	64M (default)	39m32s

As can be seen from Table 1, after optimizing the parameter SplitSize, the execution time of Mapper-side recognition and classification is 21m07s compared with that of Reducer-side recognition and classification, which is 39m32s, and the time is shortened by 46%, and the performance of the former is almost twice that of the latter. Therefore, the efficiency of library recognition can be optimized by optimizing the SplitSize in Mapper-side recognition classification.

III. C. 2) Application platform system performance testing

In order to test the performance of the classification system based on the method of this paper, tests are conducted on the application cluster with 32 nodes and the test cluster with 8 nodes to compare the performance of the system in the distributed platform system. Statistics of program execution time for running 50G, 100G, 200G, and 500G mobile Internet traffic data classification on application cluster (E1) and test cluster (E2) are shown in Fig. 7.

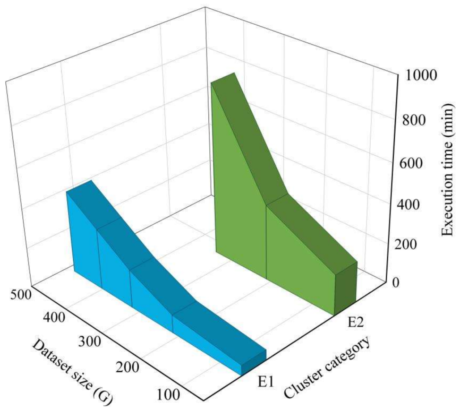

Figure 7: Performance test comparison of website classification systems

As can be seen from Fig. 7, the horizontal coordinate represents the experimental dataset size and the vertical coordinate represents the execution time of the classification task of the classification system. The two lines represent the execution time of the application cluster (E1) and the execution time of the test cluster (E2), respectively. The number of nodes in the application cluster (E1) is four times the number of nodes in the test cluster, and when processing the same task with the same amount of data, the system running time decreases with the increase of computing nodes, and the execution time of the application cluster (E1) is about 30% of the execution time of the test system. As the amount of data increases linearly, the execution time of the application cluster (E1) also increases linearly, e.g., when the data is 50G, the program execution time is 49m26s, and when the data increases to 100G, the execution time is 95m19s. When the data increases to 500G, the program execution time is 417m44s, which indicates that the classification system based on the method of this paper is more suitable for dealing with massive big data.

IV. Conclusion

This paper proposes a processing optimization method for large-scale digital media data by designing the distribution set of large-scale digital media data as well as data features, the clustering process of the items, adopting the CMMDP protocol to maintain data security, and combining the Spark-based parallel mining algorithm for big data distributed computing. In the processing platform and classification system built based on this method, not only the average execution time of the K-Means algorithm on the data is only 17.9 seconds, but also the time consumed for the data division is always less than 1.5s and the number of chunks that need to be recalculated for the task module is always less than 100 blocks under different update rates.

References

[1] Ogunwole, O., Onukwulu, E. C., Sam-Bulya, N. J., Joel, M. O., & Achumie, G. O. (2022). Optimizing automated pipelines for realtime data processing in digital media and e-commerce. International Journal of Multidisciplinary Research and Growth Evaluation, 3(1), 112-120.

[2] Ohme, J., Araujo, T., Boeschoten, L., Freelon, D., Ram, N., Reeves, B. B., & Robinson, T. N. (2024). Digital trace data collection for social media effects research: APIs, data donation, and (screen) tracking. Communication Methods and Measures, 18(2), 124-141.

[3] Stergiou, C., Psannis, K., Plageras, A. P., Ishibashi, Y., & Kim, B. G. (2018). Algorithms for efficient digital media transmission over IoT and cloud networking. Journal of Multimedia Information System.

[4] El-Seoud, S. A., El-Sofany, H. F., Abdelfattah, M., & Mohamed, R. (2017). Big Data and Cloud Computing: Trends and Challenges. International Journal of Interactive Mobile Technologies, 11(2).

[5] Noraziah, A., Fakherldin, M. A. I., Adam, K., & Majid, M. A. (2017). Big data processing in cloud computing environments. Advanced Science Letters, 23(11), 11092-11095.

[6] Singh, S. P., Nayyar, A., Kumar, R., & Sharma, A. (2019). Fog computing: from architecture to edge computing and big data processing. The Journal of Supercomputing, 75, 2070-2105.

[7] Berisha, B., Mëziu, E., & Shabani, I. (2022). Big data analytics in Cloud computing: an overview. Journal of Cloud Computing, 11(1), 24.

[8] Zhou, Z., & Zhao, L. (2020). Cloud computing model for big data processing and performance optimization of multimedia communication. Computer Communications, 160, 326-332.

[9] Wu, Z., Sun, J., Zhang, Y., Wei, Z., & Chanussot, J. (2021). Recent developments in parallel and distributed computing for remotely sensed big data processing. Proceedings of the IEEE, 109(8), 1282-1305.

[10] Shukur, H., Zeebaree, S. R., Ahmed, A. J., Zebari, R. R., Ahmed, O., Tahir, B. S. A., & Sadeeq, M. A. (2020). A State of Art: Survey for Concurrent Computation and Clustering of Parallel Computing for Distributed Systems. Journal of Applied Science and Technology Trends, 1(2), 148-154.

[11] Jonas, E., Pu, Q., Venkataraman, S., Stoica, I., & Recht, B. (2017, September). Occupy the cloud: Distributed computing for the 99%. In Proceedings of the 2017 symposium on cloud computing (pp. 445-451).

[12] Nardelli, M., Cardellini, V., Grassi, V., & Presti, F. L. (2019). Efficient operator placement for distributed data stream processing applications. IEEE Transactions on Parallel and Distributed Systems, 30(8), 1753-1767.

[13] Cao, K., Liu, Y., Meng, G., & Sun, Q. (2020). An overview on edge computing research. IEEE access, 8, 85714-85728.

[14] Noura, H., Salman, O., Chehab, A., & Couturier, R. (2019). Preserving data security in distributed fog computing. Ad Hoc Networks, 94, 101937.

[15] Zhang, J., Chen, B., Zhao, Y., Cheng, X., & Hu, F. (2018). Data security and privacy-preserving in edge computing paradigm: Survey and open issues. IEEE access, 6, 18209-18237.

[16] Kumar, P. R., Raj, P. H., & Jelciana, P. (2018). Exploring data security issues and solutions in cloud computing. Procedia Computer Science, 125, 691-697.

[17] Li, Y., Gai, K., Qiu, L., Qiu, M., & Zhao, H. (2017). Intelligent cryptography approach for secure distributed big data storage in cloud computing. Information Sciences, 387, 103-115.

[18] Jeon, S. E., Lee, S. J., & Lee, I. G. (2023). Hybrid in-network computing and distributed learning for large-scale data processing. Computer Networks, 226, 109686.

[19] El-Sayed, H., Sankar, S., Prasad, M., Puthal, D., Gupta, A., Mohanty, M., & Lin, C. T. (2017). Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment. ieee access, 6, 1706-1717.

[20] Zheng, G., Zhang, H., Li, Y., & Xi, L. (2020). 5G network-oriented hierarchical distributed cloud computing system resource optimization scheduling and allocation. Computer Communications, 164, 88-99.