INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 01, January 2024, pp : 235-241

e-ISSN : 2583-1062

Impact Factor : 5.725

EFFICIENT DISTRIBUTED CACHE MANAGEMENT IN BIG DATA PROCESSING WITH MAPREDUCE

K Ramesh Babu[1], Koya Haritha[2], Roja D[3]

[1]Asst. Professor, Dept of ECE, Chalapathi Institute of Technology, Guntur, A.P, India.

[2, 3]Asst. Professor, Dept of CSE-Data Science, Chalapathi Institute of Technology, Guntur, A.P, India.

ABSTRACT

In the realm of Big Data processing, efficient data management is paramount for achieving optimal performance. Distributed caching plays a crucial role in mitigating data transfer overhead and enhancing computation speed in MapReduce frameworks. This paper proposes a novel approach for Efficient Distributed Cache Management in Big Data Processing with MapReduce. Our solution focuses on addressing the challenges associated with large-scale data processing by leveraging a distributed cache mechanism. By strategically managing and distributing cached data across the cluster, we aim to reduce the data transfer latency and improve the overall efficiency of MapReduce jobs. Key components of our approach include intelligent cache placement, dynamic cache updating, and adaptive eviction policies. We employ intelligent algorithms to determine the optimal placement of cached data based on the computation patterns and data access frequencies. Additionally, our system dynamically updates the cache contents to ensure that the most relevant and frequently accessed data is readily available for computation. Furthermore, we introduce adaptive eviction policies to manage the cache size dynamically, ensuring that the system adapts to varying workloads and resource constraints. The proposed solution is implemented and evaluated on a real-world Big Data processing environment, demonstrating significant improvements in performance compared to traditional approaches. In summary, our Efficient Distributed Cache Management in Big Data Processing with MapReduce presents a scalable and adaptable solution to enhance the efficiency of large-scale data processing tasks. The experimental results showcase the effectiveness of our approach in reducing computation time and data transfer overhead, making it a valuable contribution to the field of Big Data analytics.

Keywords- BigData, Hadoop, MapReduce, Distributed Cache, Distribute File System, Cache Management.

1. INTRODUCTION

In the era of Big Data, processing massive datasets poses unique challenges that demand innovative solutions. MapReduce, a popular programming model for distributed computing, has emerged as a cornerstone for processing and analyzing vast amounts of data across distributed clusters. However, as datasets grow in size and complexity, efficient data management becomes a critical factor in optimizing the performance of MapReduce jobs. Distributed caching is a key technique employed to alleviate the limitations posed by data transfer overhead and slow access times in distributed computing environments. By strategically storing and retrieving frequently used data closer to computation nodes, distributed caching can significantly enhance the speed and efficiency of data processing tasks. This paper introduces a comprehensive approach for Efficient Distributed Cache Management in Big Data Processing with MapReduce. The primary goal is to address the inherent challenges associated with managing and processing large-scale datasets within a MapReduce framework. Our solution is designed to intelligently handle data placement, dynamic updates, and adaptive eviction policies to achieve optimal performance. The remainder of this paper is organized as follows: Section 2 provides an overview of related work in the field of distributed caching and its applications in Big Data processing. Section 3 outlines the challenges associated with current approaches and motivates the need for an efficient distributed cache management system. In Section 4, we present the architecture and key components of our proposed solution. Section 5 describes the implementation details and experimental setup, while Section 6 presents the results and performance evaluations. Finally, Section 7 concludes the paper, summarizing the contributions and highlighting avenues for future research in the domain of Big Data processing with MapReduce and efficient distributed cache management.

1.1 Related Work:

MapReduce, a powerful paradigm for processing and analyzing large datasets, has become a cornerstone in the field of distributed computing. The fundamental design of MapReduce involves the execution of user-defined map and reduce functions on vast clusters of commodity machines, each equipped with terabytes of data. The user's program specifies a map function that processes a key/value pair, generating a set of intermediate key/value pairs. Subsequently, a reduce function consolidates all intermediate values associated with the same intermediate key. This functional programming style allows automatic parallelization of programs, enabling them to run seamlessly on extensive clusters of commodity machines. This approach involves partitioning input data, orchestrating program execution

across a set of machines, handling machine failures, and managing inter-machine communication. The appeal of MapReduce lies in its accessibility for programmers without extensive experience in parallel and distributed systems. The abstraction provided by MapReduce simplifies the development of large-scale distributed applications. The requirement for a highly scalable and colossal cluster of commodity machines underscores the need for efficient data processing on an extensive scale. While MapReduce offers advantages in terms of parallelization and fault tolerance, it has limitations. Notably, it may not perform optimally with random reads over small files, as its optimization prioritizes sustained throughput. Additionally, sharing mutable data is challenging within the MapReduce framework. To address these challenges, researchers have explored various aspects of distributed computing and caching mechanisms. The next sections delve into the specific challenges and gaps in existing approaches, paving the way for the introduction of an efficient distributed cache management system tailored for Big Data processing with MapReduce.

2. LITERATURE SURVEY

Efficient distributed cache management in Big Data processing with MapReduce is a crucial aspect to optimize the performance of data-intensive applications. Various research studies and literature reviews have explored different techniques and approaches to address this challenge. Below is a brief literature survey highlighting some key works in this area:

"MapReduce: Simplified Data Processing on Large Clusters" (2004) by Jeffrey Dean and Sanjay Ghemawat: This seminal paper introduces the MapReduce programming model, which has become the foundation for processing large-scale data in a distributed environment. It discusses the role of distributed caching in the MapReduce framework, emphasizing the need for efficient data sharing among tasks. "Improving MapReduce Performance in Heterogeneous Environments" (2009) by Matei Zaharia et al.:The paper discusses techniques to improve the performance of MapReduce in heterogeneous environments by utilizing distributed caching effectively. It introduces the concept of locality-aware task scheduling and data placement strategies for efficient cache utilization. "Towards a High-performance MapReduce Runtime with Efficient Task Communication" (2010) by Qian Zhu et al.: The paper addresses the issue of inefficient task communication in MapReduce and proposes techniques to enhance the performance by optimizing the data transfer process. It explores the impact of caching on task communication and suggests strategies for better cache utilization. "Understanding Hadoop Performance on Microarchitectural and OS Virtualization" (2011) by Christos Kozanitis et al.:

This study investigates the performance of Hadoop, the open-source implementation of MapReduce, focusing on microarchitectural aspects and OS virtualization. The paper discusses the impact of cache management on Hadoop performance and suggests optimizations to enhance efficiency.

3. DISTRIBUTED CACHE

3.1 System Overview: The distributed cache system operates within the MapReduce framework, where Map invocations are distributed across multiple machines by automatically partitioning the input data into a set of M splits. These input splits can be processed concurrently by different machines. Similarly, Reduce invocations are distributed by partitioning the intermediate key space into R pieces using a user-specified partitioning function (e.g., hash(key) mod R). Upon invoking the MapReduce function in the user program, the following sequence of actions is initiated:

3.1.1 Input Data Splitting: The MapReduce library initially splits the input files into M pieces, typically ranging from 16 megabytes to 64 megabytes per piece. The user can control this size through an optional parameter. Replications of the program are then started on a cluster of machines.

3.1.2 Master-Worker Assignment: Within the cluster, there are both master and worker nodes. The master is responsible for assigning tasks. M map tasks and R reduce tasks are to be assigned. Idle workers are selected by the master, and each is assigned either a map task or a reduce task.

3.1.3 Map Task Execution: A worker assigned a map task reads the contents of the corresponding input split. It extracts key/value pairs from the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory.

3.1.4 Intermediate Data Writing: Periodically, the buffered pairs are written to the local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are then communicated back to the master, who is responsible for forwarding these locations to the reduce workers.

3.1.5 Reduce Task Execution: When a reduce worker is informed by the master about these locations, it utilizes remote procedure calls to read the buffered data from the local disks of the map workers. Once a reduce worker has read all intermediate data, it sorts it by the intermediate keys to group occurrences of the same key together. If the amount of intermediate data exceeds memory capacity, an external sort is employed.

3.1.6 Reduce Function Application: The reduce worker iterates over the sorted intermediate data. For each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is then appended to a final output file for this specific reduces partition.

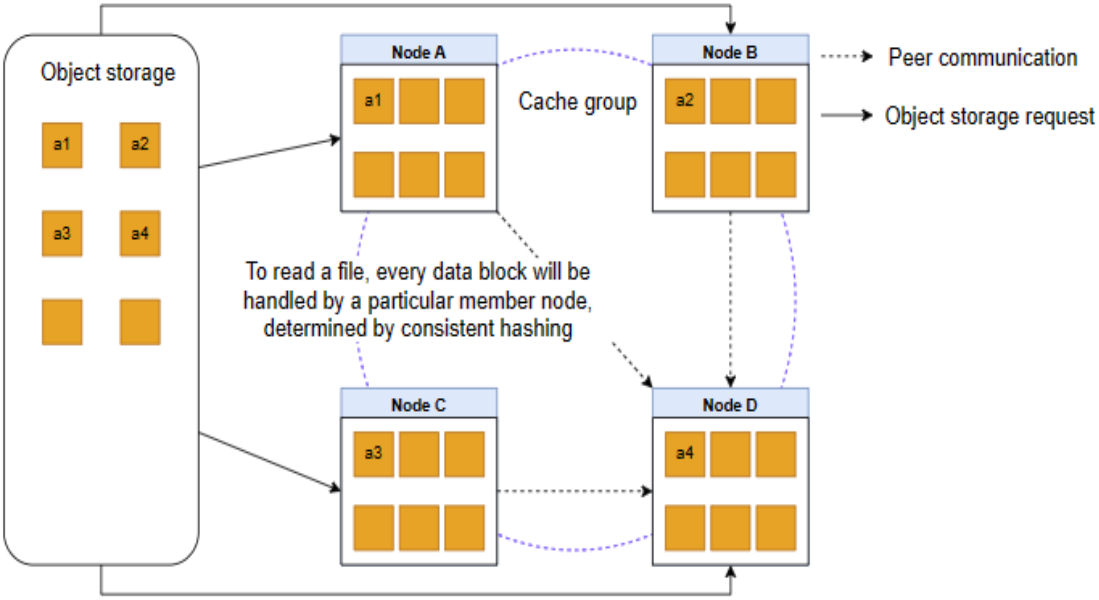

Figure. 1. Architecture of Distributed Cache

3.2 Cache Manager: The Cache Manager plays a pivotal role in storing intermediate results, particularly output (k1, v2) pairs that often serve as input for subsequent MapReduce tasks. The Cache Manager is structured with two essential phases:

3.2.1 Cache Description Scheme: Each data object is indexed based on its content. The scheme involves categorizing cache items by their source input and the operations applied to generate them.

3.2.2 Cache Request and Reply Protocol: This phase ensures efficient collation of cache items with the worker processes that require the data. The protocol minimizes transmission delays and overhead by identifying the source input, the operations applied, and the worker processes potentially in need of the cached data.

In the reduce phase, a mechanism is devised to consider the partition operations applied during the map phase. Additionally, a method is presented for reducers to leverage cached results from the map phase, thereby accelerating their execution.

3.2.3 Master and Workers: Code for the system is typically written in Java, although compatibility with other languages is facilitated through the Hadoop Streaming API. The system comprises two fundamental pieces:

3.2.4 Map Step: The master node takes a large problem input, divides it into smaller sub-problems, and distributes these to worker nodes. This process may be repeated, leading to a multi-level tree structure where worker processes handle smaller problems and return the results to the master.

3.2.5 Reduce Step: The master node combines the answers to sub-problems in a predefined way to obtain the final output/answer to the original problem.

The Distributed Cache system is organized into several modules, including:

Application to process

Map Phase Cache Description scheme

Reduce Phase Cache Description Scheme

Cache Request and Reply protocol

LRU Algorithm

B. Applications to Process

Hadoop, running in pseudo-distributed mode on a server with specific configurations, is used to process applications. The performance of Dache, compared to the traditional Hadoop (MapReduce model), is benchmarked using word-count and tera-sort requests. Word-count involves counting unique words in large input text files and is I/O-intensive. Tera-sort, on the other hand, requires varied workloads, involving both loading/storing data and a computation-intensive sorting phase.

3.3 Map Phase Cache Description: Cached data is stored in a Distributed File System (DFS), and each cache item is described by a 2-tuple: {Origin, Operation}. Origin refers to a file in the DFS, and Operation is a linear list of applied procedures on the base file. Cache descriptions can be recursive to represent sequential processing on the same dataset. For example, a cache item might be defined as {word catalog 08012012.txt, item count}, where "item count" corresponds to the word count operation on the specified data file. This structured approach facilitates efficient caching and retrieval during the MapReduce process.

4. RESULT ANALYSIS

The performance of the proposed model is assessed through the speedup and completion time of the word-count program. Figure 2 illustrates the combined presentation of completion time and speedup, considering varying sizes of appended data as a percentage of the original input file size. As observed, the speedup experiences a decrease with the growing size of appended data. However, it is noteworthy that the Distributed Cache consistently outperforms Hadoop in terms of job completion time across all scenarios. provides insights into the CPU utilization ratio of the word-count problem. The ratio is calculated by averaging the CPU utilization ratio of the MapReduce job processes over time.

The collective analysis of Figures 2 and 3 affirms that the Distributed Cache (Dache) effectively eliminates redundant tasks in incremental MapReduce jobs, resulting in a reduction in job completion time. This performance improvement is crucial in enhancing the efficiency of large-scale data processing tasks within a distributed computing environment.

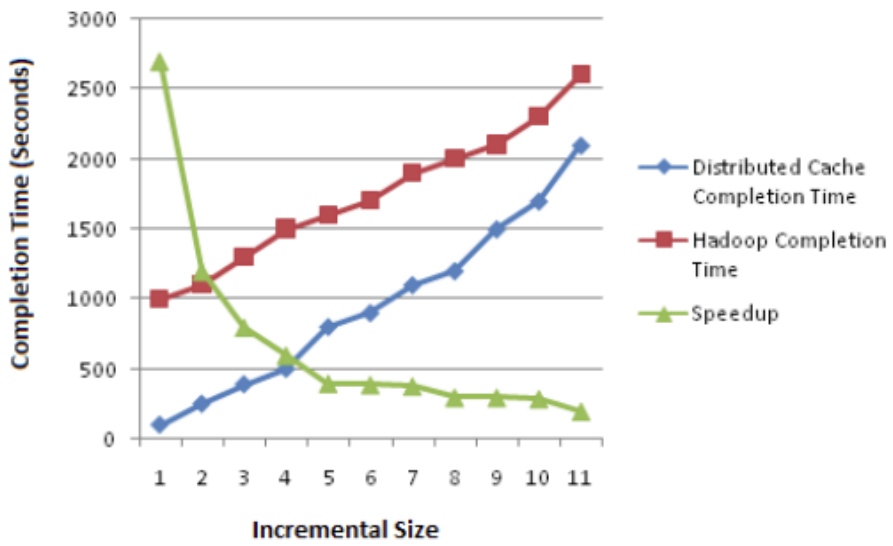

Figure 2: Completion time of Distributed Cache and Hadoop

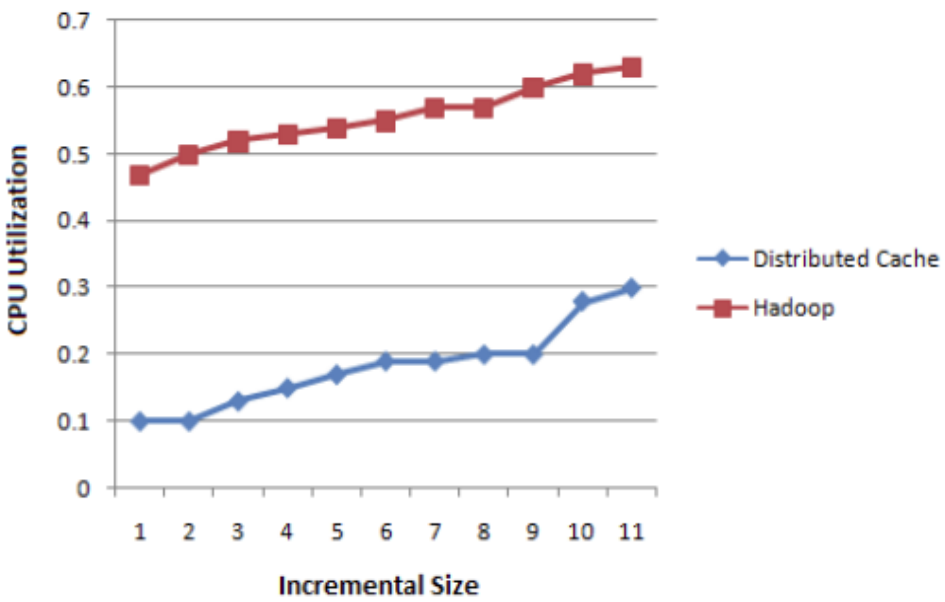

Figure . 3: CPU Utilization ratio of Distributed cache and Hadoop

5. CONCLUSION

The Distributed Cache, a data-aware caching framework tailored for MapReduce. The design and evaluation of Distributed Cache reveal that it necessitates only minor modifications in the input format and task management of the MapReduce framework. Furthermore, applications using Distributed Cache require only slight adjustments to leverage its capabilities. Testbed experiments conducted during the evaluation phase demonstrate that Distributed Cache effectively eliminates duplicate tasks in incremental MapReduce jobs. Moreover, it is noteworthy that the implementation of Distributed Cache does not demand substantial changes to the application code, simplifying its integration into existing MapReduce workflows.

Looking ahead, future work could focus on refining algorithms to achieve even more accurate CPU execution. The continuous improvement and optimization of Distributed Cache algorithms can further contribute to the efficiency and performance of large-scale data processing tasks within the MapReduce paradigm. Overall, Distributed Cache emerges as a promising solution with the potential to streamline data-aware caching in MapReduce frameworks, paving the way for enhanced performance and reduced redundancy in computational tasks.

5.1 Future Scope: The successful implementation and evaluation of Distributed Cache in the context of MapReduce open avenues for future research and enhancements. Several potential areas of focus include:

5.1.1 Algorithm Refinement: Further research can be directed towards refining and optimizing the algorithms employed by Distributed Cache. Fine-tuning these algorithms can lead to even more accurate CPU execution, enhancing overall performance.

5.1.2 Scalability: Investigating the scalability of Distributed Cache for even larger datasets and clusters is crucial. Understanding how well the framework performs as the scale of data processing increases is essential for its applicability in diverse big data scenarios.

5.1.3 Dynamic Adaptability: Enhancing the system's ability to dynamically adapt to varying workloads and changing resource constraints would be beneficial. This could involve developing adaptive mechanisms that adjust cache management strategies based on real-time conditions.

6. REFERENCES

[1] Kommineni, K. K. ., & Prasad, A. . (2023). A Review on Privacy and Security Improvement Mechanisms in MANETs. International Journal of Intelligent Systems and Applications in Engineering, 12(2), 90–99. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/4224

[2] Vellela, S.S., Balamanigandan, R. Optimized clustering routing framework to maintain the optimal energy status in the wsn mobile cloud environment. Multimed Tools Appl (2023). https://doi.org/10.1007/s11042-023-15926-5

[3] Vellela, S. S., Reddy, B. V., Chaitanya, K. K., & Rao, M. V. (2023, January). An Integrated Approach to Improve E-Healthcare System using Dynamic Cloud Computing Platform. In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 776-782). IEEE.

[4] K. N. Rao, B. R. Gandhi, M. V. Rao, S. Javvadi, S. S. Vellela and S. Khader Basha, "Prediction and Classification of Alzheimer's Disease using Machine Learning Techniques in 3D MR Images," 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 2023, pp. 85-90, doi: 10.1109/ICSCSS57650.2023.10169550.

[5] VenkateswaraRao, M., Vellela, S., Reddy, V., Vullam, N., Sk, K. B., & Roja, D. (2023, March). Credit Investigation and Comprehensive Risk Management System based Big Data Analytics in Commercial Banking. In 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS) (Vol. 1, pp. 2387-2391). IEEE [6]

[6] S Phani Praveen, RajeswariNakka, AnuradhaChokka, VenkataNagarajuThatha, SaiSrinivasVellela and UddagiriSirisha, "A Novel Classification Approach for Grape Leaf Disease Detection Based on Different Attention Deep Learning Techniques" International Journal of Advanced Computer Science and Applications(IJACSA), 14(6), 2023. http://dx.doi.org/10.14569/IJACSA.2023.01406128

[7] Vellela, S. S., & Balamanigandan, R. (2022, December). Design of Hybrid Authentication Protocol for High Secure Applications in Cloud Environments. In 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS) (pp. 408-414). IEEE.

[8] Vullam, N., Vellela, S. S., Reddy, V., Rao, M. V., SK, K. B., & Roja, D. (2023, May). Multi-Agent Personalized Recommendation System in E-Commerce based on User. In 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC) (pp. 1194-1199). IEEE.

[9] Vellela, S. S., Balamanigandan, R., & Praveen, S. P. (2022). Strategic Survey on Security and Privacy Methods of Cloud Computing Environment. Journal of Next Generation Technology (ISSN: 2583-021X), 2(1).

[10] Vellela, S. S., & Krishna, A. M. (2020). On Board Artificial Intelligence With Service Aggregation for Edge Computing in Industrial Applications. Journal of Critical Reviews, 7(07), 2020.

[11] Madhuri, A., Jyothi, V. E., Praveen, S. P., Sindhura, S., Srinivas, V. S., & Kumar, D. L. S. (2022). A New Multi-Level Semi-Supervised Learning Approach for Network Intrusion Detection System Based on the 'GOA'. Journal of Interconnection Networks, 2143047.

[12] Madhuri, A., Praveen, S. P., Kumar, D. L. S., Sindhura, S., &Vellela, S. S. (2021). Challenges and issues of data analytics in emerging scenarios for big data, cloud and image mining. Annals of the Romanian Society for Cell Biology, 412-423.

[13] Praveen, S. P., Sarala, P., Kumar, T. K. M., Manuri, S. G., Srinivas, V. S., &Swapna, D. (2022, November). An Adaptive Load Balancing Technique for Multi SDN Controllers.In 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS) (pp. 1403-1409).IEEE.

[14] Vellela, S. S., Basha Sk, K., & Yakubreddy, K. (2023). Cloud-hosted concept-hierarchy flex-based infringement checking system. International Advanced Research Journal in Science, Engineering and Technology, 10(3).

[15] Rao, M. V., Vellela, S. S., Sk, K. B., Venkateswara, R. B., & Roja, D. (2023). SYSTEMATIC REVIEW ON SOFTWARE APPLICATION UNDERDISTRIBUTED DENIAL OF SERVICE ATTACKS FOR GROUP WEBSITES. Dogo Rangsang Research Journal UGC Care Group I Journal, 13(3), 2347-7180.

[16] Venkateswara Reddy, B., Vellela, S. S., Sk, K. B., Roja, D., Yakubreddy, K., & Rao, M. V. Conceptual Hierarchies for Efficient Query Results Navigation. International Journal of All Research Education and Scientific Methods (IJARESM), ISSN, 2455-6211.

[17] Sk, K. B., Roja, D., Priya, S. S., Dalavi, L., Vellela, S. S., & Reddy, V. (2023, March). Coronary Heart Disease Prediction and Classification using Hybrid Machine Learning Algorithms. In 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA) (pp. 1-7). IEEE.

[18] Sk, K. B., & Vellela, S. S. (2019). Diamond Search by Using Block Matching Algorithm. DIAMOND SEARCH BY USING BLOCK MATCHING ALGORITHM", International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN, 2349-5162.

[19] Yakubreddy, K., Vellela, S. S., Sk, K. B., Reddy, V., & Roja, D. (2023). Grape CS-ML Database-Informed Methods for Contemporary Vineyard Management. International Research Journal of Modernization in Engineering Technology and Science, 5(03).

[20] Vellela, Sai Srinivas and Chaganti, Aswini and Gadde, Srimadhuri and Bachina, Padmapriya and Karre, Rohiwalter, A Novel Approach for Detecting Automated Spammers in Twitter (June 24, 2023). Mukt Shabd Journal Volume XI, Issue VI, JUNE/2022 ISSN NO : 2347-3150, pp. 49-53 , Available at SSRN: https://ssrn.com/abstract=4490635

[21] Vellela, Sai Srinivas and Pushpalatha, D and Sarathkumar, G and Kavitha, C.H. and Harshithkumar, D, ADVANCED INTELLIGENCE HEALTH INSURANCE COST PREDICTION USING RANDOM FOREST (March 1, 2023). ZKG International, Volume VIII Issue I MARCH 2023, Available at SSRN: https://ssrn.com/abstract=4473700

[22] D, Roja and Dalavai, Lavanya and Javvadi, Sravanthi and Sk, Khader Basha and Vellela, Sai Srinivas and B, Venkateswara Reddy and Vullam, Nagagopiraju, Computerised Image Processing and Pattern Recognition by Using Machine Algorithms (April 10, 2023). TIJER International Research Journal, Volume 10 Issue 4, April 2023, Available at SSRN: https://ssrn.com/abstract=4428667

[23] Vellela, Sai Srinivas and Basha Sk, Khader and B, Venkateswara Reddy and D, Roja and Javvadi, Sravanthi, MOBILE RFID APPLICATIONS IN LOCATION BASED SERVICES ZONE (June 14, 2023). International Journal of Emerging Technologies and Innovative Research, Vol.10, Issue 6, page no. ppd851-d859, June2023, http://www.jetir.org/papers/JETIR2306410.pdf

[24] Vellela, Sai Srinivas and Sk, Khader Basha and B, Venkateswara Reddy, Cryonics on the Way to Raising the Dead Using Nanotechnology (June 18, 2023). INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS), Vol. 03, Issue 06, June 2023, pp : 253-257,

[25] Vellela, Sai Srinivas and D, Roja and B, Venkateswara Reddy and Sk, Khader Basha and Rao, Dr M Venkateswara, A New Computer-Based Brain Fingerprinting Technology (June 18, 2023). International Journal Of Progressive Research In Engineering Management And Science, Vol. 03, Issue 06, June 2023, pp : 247-252 e-ISSN : 2583-1062.,

[26] Gajjala, Buchibabu and Mutyala, Venubabu and Vellela, Sai Srinivas and Pratap, V. Krishna, Efficient Key Generation for Multicast Groups Based on Secret Sharing (June 22, 2011). International Journal of Engineering Research and Applications,Vol. 1, Issue 4, pp.1702-1707, ISSN: 2248-9622

[27] Kiran Kumar Kommineni, Ratna Babu Pilli, K. Tejaswi, P. Venkata Siva,Attention-based Bayesian inferential imagery captioning maker,Materials Today: Proceedings,2023,ISSN 2214-7853,https://doi.org/10.1016/j.matpr.2023.05.231.

[28] Venkateswara Reddy, B., &KhaderBashaSk, R. D. Qos-Aware Video Streaming Based Admission Control And Scheduling For Video Transcoding In Cloud Computing. In International Conference on Automation, Computing and Renewable Systems (ICACRS 2022).

[29] Reddy, N.V.R.S., Chitteti, C., Yesupadam, S., Desanamukula, V.S., Vellela, S.S., Bommagani, N.J. (2023). Enhanced speckle noise reduction in breast cancer ultrasound imagery using a hybrid deep learning model. Ingénierie de Systèmesd'Information, Vol. 28, No. 4, pp. 1063-1071. https://doi.org/10.18280/isi.280426

[30] Vellela, S.S., Balamanigandan, R. An intelligent sleep-awake energy management system for wireless sensor network. Peer-to-Peer Netw. Appl. (2023). https://doi.org/10.1007/s12083-023-01558-x

[31] Rao, D. M. V., Vellela, S. S., Sk, K. B., &Dalavai, L. (2023). Stematic Review on Software Application Under-distributed Denial of Service Attacks for Group Website. DogoRangsang Research Journal, UGC Care Group I Journal, 13.

[32] S. S. Priya, S. SrinivasVellela, V. R. B, S. Javvadi, K. B. Sk and R. D, "Design And Implementation of An Integrated IOT Blockchain Framework for Drone Communication," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-5, doi: 10.1109/CONIT59222.2023.10205659.

[33] N. Vullam, K. Yakubreddy, S. S. Vellela, K. BashaSk, V. R. B and S. SanthiPriya, "Prediction And Analysis Using A Hybrid Model For Stock Market," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-5, doi: 10.1109/CONIT59222.2023.10205638.

[34] K. K. Kumar, S. G. B. Kumar, S. G. R. Rao and S. S. J. Sydulu, "Safe and high secured ranked keyword searchover an outsourced cloud data," 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, India, 2017, pp. 20-25, doi: 10.1109/ICICI.2017.8365348.

[35] Sk, K. B., Vellela, S. S., Yakubreddy, K., &Rao, M. V. (2023). Novel and Secure Protocol for Trusted Wireless Ad-hoc Network Creation. KhaderBashaSk, Venkateswara Reddy B, SaiSrinivasVellela, KancharakuntYakub Reddy, M VenkateswaraRao, Novel and Secure Protocol for Trusted Wireless Ad-hoc Network Creation, 10(3).

[36] Vellela, S. S., Sk, K. B., Dalavai, L., Javvadi, S., &Rao, D. M. V. (2023). Introducing the Nano Cars Into the Robotics for the Realistic Movements. International Journal of Progressive Research in Engineering Management and Science (IJPREMS) Vol, 3, 235-240.

[37] Kumar, K. & Babu, B. & Rekha, Y.. (2015). Leverage your data efficiently: Following new trends of information and data security. International Journal of Applied Engineering Research. 10. 33415-33418.

[38] S. S. Vellela, V. L. Reddy, R. D, G. R. Rao, K. B. Sk and K. K. Kumar, "A Cloud-Based Smart IoT Platform for Personalized Healthcare Data Gathering and Monitoring System," 2023 3rd Asian Conference on Innovation in Technology (ASIANCON), Ravet IN, India, 2023, pp. 1-5, doi: 10.1109/ASIANCON58793.2023.10270407.