Vol 13, Issue 06, JUNE / 2022

ISSN NO:0377-9254

Journal of Engineering Sciences

Efficiency Improvement of Hadoop MapReduce by Integrating Mapper Data

A Ravi Kishore [1], Vipparla Naveen Kumar [2]

[1,2] Asst. Professor, Bapatla Engineering College

[1]rawikishorre.arangi@becbapatla.ac.in, [2]naveenvipparla@gmail.com

Abstract- Hadoop is a platform for storing and processing big scale data. Hadoop Distributed File System (HDFS) handles the storage component and MapReduce conducts the data processing while Yet Another Resource Negotiator(YARN) maintains all the resources of the cluster. MapReduce is a programming framework for user specified Mapping and Reducing. Key-value pairs created by Mapper are unfiltered and repeating, sent to Reducer results in a significant needless data throughput producing bandwidth overdraw. In this article, a novel approach called Inner Map Combining(IMC) for Map phase is introduced. The values of repeating keys get blended using this process within the Mapper. A successful testing was done in 12 distinct methods to assess the effectiveness of the algorithm. The test demonstrates that IMC done using Default Combiner(DC) of MapReduce is 65 percent more efficient than basic MapReduce program without any combiner. This approach may be followed with MapReduce programming for substantially quicker calculation.

Keywords— Hadoop, MapReduce, Inner Map Combiner, Combiner and Performance Enhancement.

I. INTRODUCTION

The internet revolution of the 21st century will be chronicled in history because of its reliance on internet connection. Every aspect of our everyday lives is now linked to the internet in order to make our lives more convenient. However, on the other hand, social media data such as Facebook posts and Instagram photos, corporate cloud data such as ERP and IOT data, photos and videos from IoT devices, emails and medical records, etc. are all creating a lot of data. Compared to prior years, the pace of growth has been exponential. The vast amount of information may be broken down into two groups. Data that has been created at an exponential rate of velocity and volume from a variety of sources is known as big data. In the world of big data, Hadoop is the most popular and adaptable open source computing paradigm. One of Hadoop's most important features is its programming framework, called MapReduce. This two-part MapReduce model [1] is the foundation of the whole framework. It's up to the user to customise this function to their individual requirements. i. Map Key, value pairs from Input split are used as input for this function, which returns intermediate key, value pairs. Also authored by the user, this reducer receives the intermediate key and the related values for that key and merges them together to produce a smaller collection of values. iii. Those tiny programmes that make up MapReduce are called MapReduce components. Input, Map, Reduce, and Output make up the system. There is no filtering of any kind in the current system after the Map Phase. As there may be repeating (key, value) pairs, this adds extra bandwidth requirements for data transit from Map to Reduce. The ultimate purpose of this project is to conduct a detailed investigation. To improve Hadoop MapReduce's performance, identify the weak points and devise a strategy. Repeating keys are prevented by using HashMaps to store each created pair of keys and comparing each new key to an existing key. In the end, putting into practise the technique that was recommended. Observe the results and draw conclusions regarding the method's effectiveness. The remainder of the document is structured as follows: Section I is devoted to presenting the results of an extensive investigation. The suggested model of the system is presented in Section I, Section 1. In Section IV, we'll talk about how things went and how they turned out. Finally, section V wraps things off.

II. RELATEDWORKS

A lot of effort has been done to improve Hadoop's performance in this region, and the literature will be studied. Researchers, academics and developers have all contributed to the field. HDFS is the Hadoop cluster's storage management system. There has been a lot of effort put towards improving HDFS's performance. Small file processing was the focus of research by Yang Zhang et al. [2]. Multi-pipeline data transmission has been made possible by Zhang, H. et al. [3]. A dynamic Data Placement technique was devised by Chia-Wei Lee et al. [4]. A multi-dimensional indexing approach based on R trees was presented by Liao, H. et al. [5]. In order to manage its resources, Hadoop relies on YARN [6]. This layer of Hadoop is critical to the cluster's overall performance. Shuffle

Journal of Engineering Sciences

in Hadoop has been introduced as a standalone service by Guo, Y. Bypassing the JVM, Lu, X. and coworkers [8] have developed a buffer management technique. The HASTE scheduling technique was developed by Yao Y et al. based on task dependence and resource requirements [9]. An adaptive configuration and elastic container were developed by Ding, X. et al. MapReduce's performance is the most important factor in determining the total performance of a cluster's computation. Limited Node Block Placement Policy was used by Lee, S. et al. [11]. (LNBPP). Using the HDFS distributed cache and the resource-aware data insertion technique, Spivak, A. et al. constructed an algorithm. Htay T. et al. [14] employed a concurrent container whereas Fan, Y. et al [13] focused on the scheduling. MapReduce speed improvement has seen a lot of effort, but relatively little has been done to reduce data throughput or crucial filtering between MapReduce stages. There have been several studies using multi-level node combiners, including Jeyaraj R. et al [15], Vinutha, D. C. et al [16], and Lee, W. et al [17]. Memory cache is used to store intermediate data and combine it before the shuffle part, while the in-node combiner is used to combine all the intermediate key generated on a node. This work was motivated by previous efforts in combiner to reduce I/O throughput. After the Mapper emitted the intermediate key, all of them sought to reduce the throughput. Identifying and addressing this gap was a primary driving force for our project.

III. PROPOSED SYSTEM ARCHITECTURE

Here are some of MapReduce's shortcomings. Then, the measures that may be done to overcome them can be identified.

A. Deficiencies in the Current Model The intermediate keys are generated by the user-defined Map function and sent to the user-defined Reduce function by the MapReduce in general. Because there is no filter to check for duplicated or repetitive keys, the present mapping method has a major flaw. Keys that are repeated in algorithm 1 of the basic mapper do not have any special criteria. These reoccurring keys, as seen in figure 1, result in excessive data transmission. More time will be wasted as a consequence. As a result of the mapper function, each node has a default combination method in place that is a reducer for reducing key/value pairs. Mapper output must be minimised so that repeating keys may be handled before they are sent to storage, hence this is the focus of the project.

B. An Internal Map Combiner Model Is Being Developed.

To see whether a key has already been produced by the mappers, one option is on apply a filter to the data. Mapper intermediate keys are identified by using a filter. When a key is recognised as recurring, the linked key's value is added to the original value. Figure 2 shows how the number of created intermediate key value pairs will be significantly reduced after all of the mappers have completed their mapping in this manner. The intermediate key created by the proposed mapper is stored in a HashMap before the output is broadcast. MapReduce's suggested mapper function has a technique described in algorithm 2 that may locate and combine the values of recurrent intermediate keys inside the mapper. The suggested mapper methodology's logical flow is shown graphically in the flowchart 3.

C. *Framework of the Suggested Design*

It is indicated in the study that there are two ways to combine data. After the mapper has done creating the intermediate key-value pairs, the DC or a tiny reduction has been shown to be particularly efficient at lowering the I/O throughput of the mapper [18]. With the new IMC, key-value pairs are combined within the mapper immediately after they are formed and before they are written to distributed storage, making it faster and more efficient. Each mapper function generates a unique set of key-value pairs in this manner. Because of this, the suggested strategy is more time-saving. In the next part, we will illustrate the theory and the importance of the suggested system model via implementation and performance analysis.

JES
Journal of Engineering Sciences

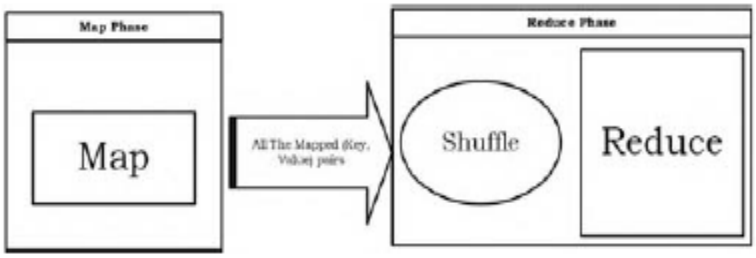

Fig. 1. Limitations of MapReduce

Algorithm 1: The Map Function

1 **for** *each element* m_{ij} *of* M **do**
2 produce $(key, value)$ pairs as $((i,k),(M,j,m_{ij}))$, for $k = 1,2,3,..$ up to the number of columns of N
3 **for** *each element* n_{jk} *of* N **do**
4 produce $(key, value)$ pairs as $((i,k),(N,j,n_{jk}))$, for $i = 1,2,3,...$ up to the number of rows of M
5 **return** *Set of (key, value) pairs that each key,* (i,k)*, has a list with values* (M,j,m_{ij}) *and* (N,j,n_{jk}) *for all possible values of* j

Algorithm 2: The Reduce Function

1 **for** *each key (i,k)* **do**
2 sort values begin with M by j in $list_M$
3 sort values begin with N by j in $list_N$
4 multiply m_{ij} and n_{jk} for j_{th} value of each list
5 sum up $m_{ij} * n_{jk}$
6 **return** $(i,k), \sum_{j=1} m_{ij} * n_{jk}$

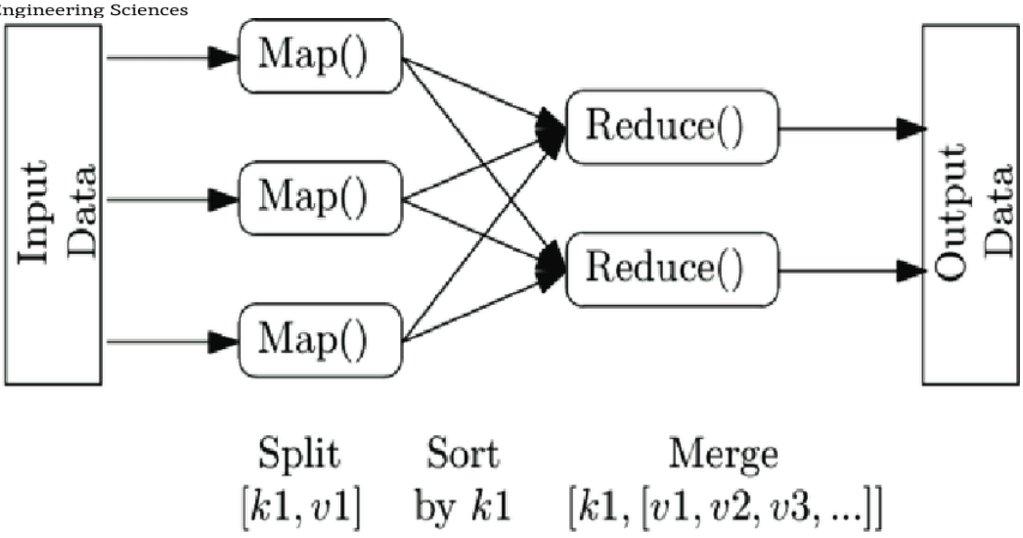

Fig. 2. MapReduce: A Proposed Model

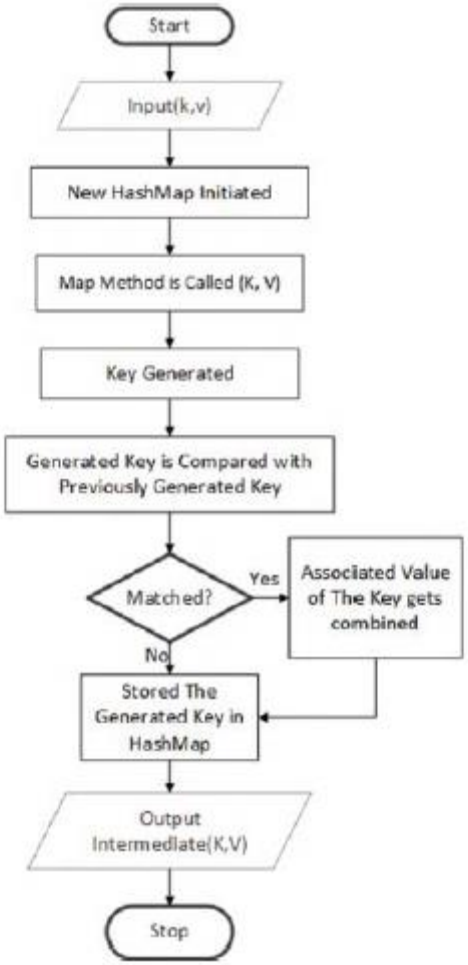

Fig. 3. The system flow of Proposed Mapper with HashMap Array

IV. RESULTS AND DISCUSSION

This section's primary objective is to prevent any changes to the current Hadoop features and to maximise the performance gains offered by the proposed system paradigm.

A. *Setting up a test system*
A single-node Hadoop cluster was used as a testbed. An 8-core AMD Bulldozer FX-8350 CPU and 8 GB of RAM powered the node. The SATA port on the node provided 6 Gbps of data speed. Windows 10 Pro for Workstation 2004 was the operating system. Pseudo-distributed Hadoop 3.3.0 running on Java 1.8.226 was implemented. All options were left at their factory settings.

B. Workloads and Data Collection
Data in table I was created by copying and pasting tens of thousands of phrases from the internet, resulting in three distinct sets of text. In terms of picking sentences, the data sets are completely distinct from one another.

C. Testing the Code
The algorithm was tested in four different ways with three different sets of data. We used the Hadoop Cluster's Resource management job information to get an idea of how long it took to complete each task. Four

different kinds of word count programmes were used to evaluate the data sets. Traditional Word Count (WC) without the use of combiners. That's our suggested method, WC with IMC. The DC in a Traditional WC. The most comprehensive Word Counting solution, combining IMC and DC. There were three separate pieces of data. In the case of a single file or data set, Containing two distinct text files or two times the amount of data The maximum number of text files or data files that may be used is three.

TABLE I
INFORMATION ABOUT GENERATED DATASETS

Dataset	Size(KB)	no. of Words
textinput	152,168	163,855
textinput2	168,320	131,072
textinput3	162,816	131,073

TABLE II
TEST RESULTS OF THE DIFFERENT PROGRAM SHOWING THE JOB COMPLETION TIME WITH DIFFERENT WORKLOAD

Name of Program	1x Data	2x Data	3x Data
Traditional	1 min 12 sec	1 min 43 sec	2 min 8 sec
Inner Map Combiner	57 sec	1 min 5 sec	1 min 25 sec
Default Combiner	41 sec	42 sec	54 sec
Inner Map Combiner with Default Combiner	34 sec	36 sec	45 sec

The findings of these tests have been surprising and startling, to say the least. The results are shown in Table II. During the evaluation, just the algorithm's timing performance was examined; memory and storage needs were not examined.

D. Analyzing the Data

Figure 4 depicts the test findings as a visual representation based on the successful testing and investigations.

TABLE III
JOB COMPLETION TIME FOR INNER MAP COMBINER VS TRADITIONAL WORD COUNT IN DIFFERENT DATA LOAD

Program	1x data	2x data	3x data
Traditional Word Count	72 sec	103 sec	128 sec
Inner Map Combiner	57 sec	65 sec	85 sec
Efficiency	20.83%	36.89%	33.59%

TABLE IV
JOB COMPLETION TIME FOR DEFAULT COMBINER VS INNER MAP COMBER PLUS DEFAULT COMBINER IN DIFFERENT DATA LOAD

Program	1x data	2x data	3x data
Default combiner	41 sec	42 sec	54 sec
Inner Map Combiner plus Default combiner	34 sec	36 sec	45 sec
Efficiency	17.07%	14.29%	16.66%

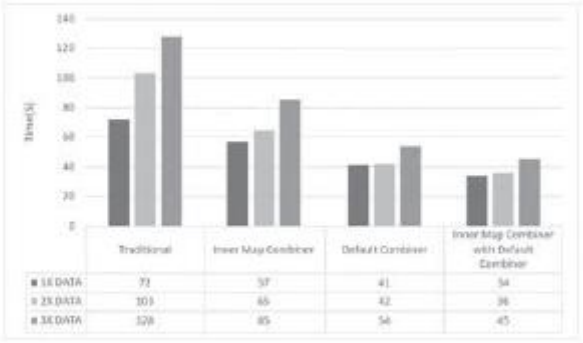

Fig. 4. Visual Representation of Test Completion Time

Table III compares IMC's efficiency to that of Traditional WC, while Table IV compares IMC&DC to only DC under various data loads. The test results support these conclusions. Figure 5 shows the time (in milliseconds) required by Mapper and Reducer to accomplish the work using various word count programmes with a 1x data load, 6 shows a 2x data load, and 7 shows a 3x data load.

*1) Analysis's conclusions:*Figure 8 depicts the overall performance boost that may be achieved by reducing the time it takes to accomplish various tasks.

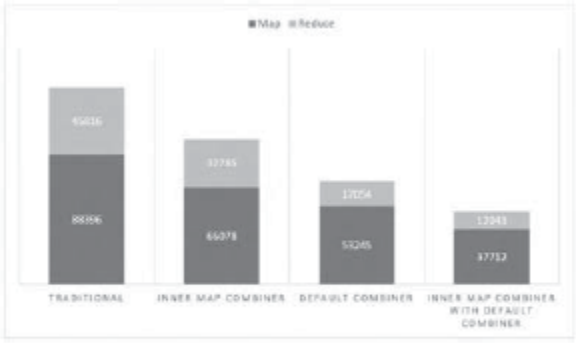

Fig. 5. Performance of Mapper and Reducer for 1x Data Load

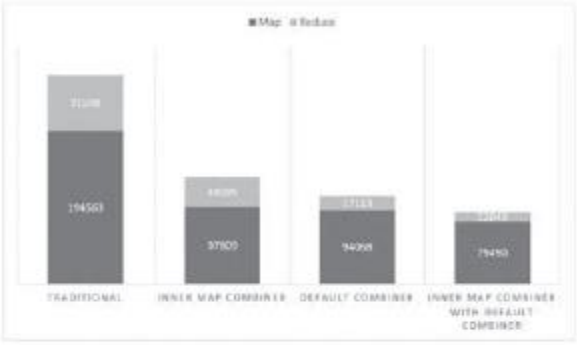

Fig. 6. Job completion Time of Mapper and Reducer for 2x Data Load

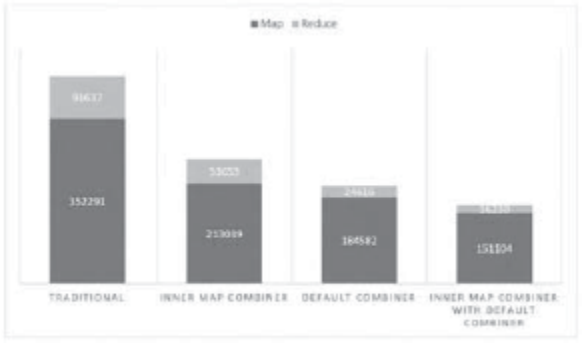

Fig. 7. Job completion Time of Mapper and Reducer for 3x Data Load

TABLE V
INNER MAP COMBINER PLUS DEFAULT COMBINER VS TRADITIONAL. WORD COUNT IN 3X DATA LOAD

Program	Completion Time
Traditional	128 sec
Inner Map Combiner with Default combiner	45 sec
Efficiency	64.84%

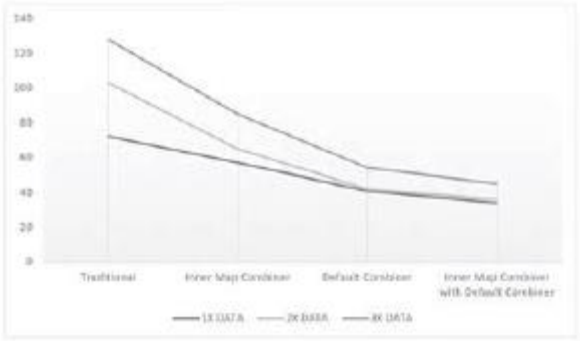

Fig. 8. Visual Representation of Performance Improvements

Figures 5, 6, and 7 illustrate that the reducer becomes more efficient as the quantity of data increases. Testing showed that IMC was able to increase efficiency by 36 percent compared to a DC system with no combiner, and by 17 percent when IMC was applied with the DC. In all, IMC is over 65 percent more efficient than standard MapReduce when implemented with DC, as seen in table V. In order to avoid comparing IMC to DC, we must remember that DC is a reduced implementation of the real reducer, whereas IMC is a filtering function.

V. FUTURE SCOPE AND CONCLUSION

The foundation of this project is a thorough investigation on how to improve Hadoop's performance. This study proposes a novel method for merging the values of repeated keys. Inner Map Combining is the name of the new method. Mapper output is created before the data value is mapped in the map method, as the name suggests. An array called a HashMap is used in the technique. Successful experiments were done in twelve different ways to demonstrate their usefulness and efficiency. IMC was shown to be more than a third more effective than a simple combination. When IMC is used in conjunction with the DC, the MapReduce application performs 17 percent better than when using simply the Default Combiner. Nearly 65% more efficient than a standard programme, the IMC works best with DC. MapReduce and Hadoop's performance may be improved by this approach.

REFERENCES

[1] Y. Yenıaydin and S. Köklücan, "Simultaneous Localization and Mapping with One Particle," 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021, pp. 1-4, doi: 10.1109/SIU53274.2021.9478047.

[2] T. J. Son, A. H. Abu Hassan and M. H. Jairan, "Optimized Robot Mapping and Obstacle Avoidance using Stereo Vision," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2021, pp. 279-284, doi: 10.1109/ISCAIE51753.2021.9431769.

[3] Z. Wang et al., "Building a post-layout simulation performance model with global mapping model fusion technique," in Tsinghua Science and Technology, vol. 27, no. 3, pp. 512-525, June 2022, doi: 10.26599/TST.2021.9010042.

[4] A. Bekcan and H. Ergezer, "Design and Implementation of Visual Simultaneous Localization and Mapping (VSLAM) Navigation System," 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021, pp. 1-4, doi: 10.1109/SIU53274.2021.9477703.

[5] A. Yao and L. Di, "Machine Learning-based Pre-season Crop Type Mapping: A Comparative Study," 2021 9th International Conference on Agro-Geoinformatics (Agro-Geoinformatics), 2021, pp. 1-4, doi: 10.1109/Agro-Geoinformatics50104.2021.9530356.

[6] H. Jeon, D. -j. Kim and J. Kim, "Water Body Detection using Deep Learning with Sentinel-1 SAR satellite data and Land Cover Maps," 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, 2021, pp. 8495-8498, doi: 10.1109/IGARSS47720.2021.9553555.

[7] J. Wang, B. Wang, Z. Huang, H. Li, J. Wu and Y. Wen, "Equivalence Analysis of Transmission Line Ice Coating Reduced-scale Test Based on Finite Element Simulation," 2021 International Conference on Advanced Electrical Equipment and Reliable Operation (AEERO), 2021, pp. 1-4, doi: 10.1109/AEERO52475.2021.9708220.

[8] G. Liu, B. Yu, L. Huang, L. Shi, X. Gao and L. He, "Human-Interactive Mapping Method for Indoor Magnetic Based on Low-Cost MARG Sensors," in IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-10, 2021, Art no. 9505510, doi: 10.1109/TIM.2021.3052026.

[9] Y. Liu, D. Lu, B. Cai and J. Wang, "Comprehensive Probability Map-matching Method for Digital Track Map Validation," 2021 International Conference on Electromagnetics in Advanced Applications (ICEAA), 2021, pp. 305-310, doi: 10.1109/ICEAA52647.2021.9539550.

[10] R. Wang, Q. Sun, P. Tu, J. Xiao, Y. Gui and P. Wang, "Reduced-Order Aggregate Model for Large-Scale Converters With Inhomogeneous Initial Conditions in DC Microgrids," in IEEE Transactions on Energy Conversion, vol. 36, no. 3, pp. 2473-2484, Sept. 2021, doi: 10.1109/TEC.2021.3050434.

[11] J. Liu, W. Xu, B. Guo, G. Zhou and H. Zhu, "Accurate Mapping Method for UAV Photogrammetry Without Ground Control Points in the Map Projection Frame," in IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 11, pp. 9673-9681, Nov. 2021, doi: 10.1109/TGRS.2021.3052466.

[12] Z. H. Bohari, M. Isa, P. J. Soh, A. Z. Abdullah, M. F. Sulaima and M. N. M. Nasir, "A Hybrid Method of Self Organizing Maps with Statistical Feature Extraction for Accurate and Efficient Partial Discharge Recognition and Clustering," 2021 IEEE International Conference on the Properties and Applications of Dielectric Materials (ICPADM), 2021, pp. 246-249, doi: 10.1109/ICPADM49635.2021.9493942.

[13] M. Song, Y. Zhong, A. Ma, X. Xu and L. Zhang, "A Joint Spectral Unmixing and Subpixel Mapping Framework Based on Multiobjective Optimization," in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-17, 2022, Art no. 5519217, doi: 10.1109/TGRS.2021.3132610.

[14] J. Brent, B. Daniel and A. Hussein, "Examining the practicality and accuracy of Unmanned Aerial System Topographic Mapping (Drones) Compared to Traditional Topographic Mapping," 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), 2021, pp. 1-7, doi: 10.1109/ICECCME52200.2021.9591036.

[15] H. Gim, M. Jeong and S. Han, "Autonomous Navigation System with Obstacle Avoidance using 2.5D Map Generated by Point Cloud," 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021, pp. 749-752, doi: 10.23919/ICCAS52745.2021.9649862.

[16] L. -H. Hoang, M. A. Hanif and M. Shafique, "TRe-Map: Towards Reducing the Overheads of Fault-Aware Retraining of Deep Neural Networks by Merging Fault Maps," 2021 24th Euromicro Conference on Digital System Design (DSD), 2021, pp. 434-441, doi: 10.1109/DSD53832.2021.00072.

[17 Y. Wu, M. Li, X. Jiang and E. Bai, "Low Complexity Data Mapping and Detection for Jointly Mapping Generalized Spatial Modulation," 2021 6th International Conference on Communication, Image and Signal Processing (CCISP), 2021, pp. 367-371, doi: 10.1109/CCISP52774.2021.9639324.

[18] Y. Wu, M. Li, X. Jiang and E. Bai, "Low Complexity Data Mapping and Detection for Jointly Mapping Generalized Spatial Modulation," 2021 6th International Conference on Communication, Image and Signal Processing (CCISP), 2021, pp. 367-371, doi: 10.1109/CCISP52774.2021.9639324.

[19] H. Li, K. Qu and J. Zhou, "Reconstructing Sound Speed Profile From Remote Sensing Data: Nonlinear Inversion Based on Self-Organizing Map," in IEEE Access, vol. 9, pp. 109754-109762, 2021, doi: 10.1109/ACCESS.2021.3102608.

[20] W. Zhang, C. Zhanga, X. Niu and S. Fu, "Research on Robot Autonomous Mapping Technology in Complex Environment Based on ROS," 2021 IEEE 4th International Conference on Big Data and Artificial Intelligence (BDAI), 2021, pp. 75-79, doi: 10.1109/BDAI52447.2021.9515287.

[21] P. Wang, L. Wang, H. Leung and G. Zhang, "Super-Resolution Mapping Based on Spatial–Spectral Correlation for Spectral Imagery," in IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 3, pp. 2256-2268, March 2021, doi: 10.1109/TGRS.2020.3004353.

[22] A. E. Dalzotto, M. Ruaro, L. V. Erthal and F. G. Moraes, "Dynamic Mapping for Many-cores using Management Application Organization," 2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2021, pp. 1-6, doi: 10.1109/ICECS53924.2021.9665547.