EAI Endorsed Transactions
on AI and Robotics

Research Article EAI.EU

# Design and Implementation of a Novel Parallel Algorithm for Efficient Image Compression in High-Performance Computing

Hewa Majeed Zangana[1,*]

[1]IT Department, Duhok Technical College, Duhok Polytechnic University, Duhok, Iraq

## Abstract

The focus of this paper is to describe the development and the architecture of a new parallel algorithm targeted for image compression within High Performance Computing context. The suggested algorithm apply parallel processing strategies for the image data set in order to minimize the amount of computation while at the same time optimize the compression ratio and speed. When combining new parallelism strategies with newly developed and existing methods of data compression, the result is visibly better in both, compression ratio and time, as opposed to comparable existing algorithms. Experimental results performed on different HPC environment prove that the solution put forward is quite scalable and efficient; therefore, it should be considered in applications where real time image processing is crucial. The novelty of this work lies in the design and implementation of a parallel algorithm that simultaneously integrates bitplane coding, wavelet transforms, and predictive coding within a scalable HPC framework—an approach not yet addressed comprehensively in existing methods. This hybrid strategy provides significant improvements in compression ratio, execution time, and scalability when compared to state-of-the-art image compression techniques.

**Keywords:** Algorithm, High-Performance Computing, Image Compression, Parallel Processing, Scalability

Received on 14 November 2024, accepted on 17 June 2025, published on 30 June 2025



doi: 10.4108/airo.7832

## 1. Introduction

The swift increase of data produced by today's applications especially in the areas of remote sensing, medical imaging, and multimedia has posed a challenge requiring effective image compression to take care of large data amounts with as much quality as possible. HPC becomes instrumental to tackle the issues come with temperature, pressure, velocity fields to process and compress the large image data and their rapid processing and improved storage density. However, such types of compression algorithms return rather low compressibility ratios and they are insufficient for solving the modern problems of data storage and processing, especially in the context of their introduction in HPC with the help of parallel computations.

The adoption of parallel processing techniques in image compression algorithms has received interest considering the improvement of efficiency. Earlier, researchers have also proposed quite a few related methods for parallel image compression, which have shown enhanced performance in respect to speed and compression ratios. For example, implementations based on GPUs have been described and reported to improve the rate of compression tasks, as could be observed in the work done by (1) for the introduction of a GPU implementation of bit plane coding for the high-performance image compression. In the same way, (2) also compared various HPC platforms for hyperspectral imaging with an emphasis to parallel processing to enables improved results.

*Corresponding author. Email: hewa.zangana@dpu.edu.krd

This is further supported by development in other fields, for example; deep learning, and big data which make the other algorithms needed in scalable image compression. In their recent paper (3), have highlighted the potential for deep-learning–based multimedia applications together with high-performance computing facilities and noted the need for effective compression in these contexts. Furthermore, enhancement of new parallel algorithms has been proven to enhance the performance of hyperspectral image processing as identified by (4) for progress of parallel techniques.

Thus, while there have been significant recent advances in many areas of parallel computing for image processing, there is still a lack of detailed studies about parallel algorithms for image compression, especially as applied to high performance computing environments. Following the latest work of (5,6), the essential aspects of parallelism in the approaches to geodynamic numerical simulations and FPGA-based image compression are revealed. Said this, to the best of our knowledge, at present, a global strategy that offers a full use of the novel opportunities of modern generation HPC infrastructures for the image compression is still missing.

In this paper, we present a novel parallel algorithm for image compression to be used in the High-Performance Computing environments. They include the parallel processing technology and HPC in which the current advancement was integrated to enhance the proportionality of other ratios and time consumed by the process. In this work, the existing approaches have been critiqued and new strategies for parallelism have been introduced to improve the methods of image compression for the large-scale applications.

This paper makes the following contributions:

- Proposes a novel parallel image compression algorithm integrating multiple compression techniques adaptable to image characteristics.
- Implements the algorithm on modern HPC platforms using CUDA and OpenCL, leveraging both GPU and FPGA acceleration.
- Evaluates the performance through comprehensive benchmarking on multiple image sizes and system configurations.
- Demonstrates scalability and superior performance over traditional and existing parallel algorithms in terms of compression ratio, processing time, and image quality (PSNR, SSIM).

## 2. Literature Review

HPC technology has experienced a lot of development in the recent past especially in the image compression domain where parallelism is gradually being incorporated to enhance the efficiency. Thus, this literature review describes the main achievements and recent advances in the field, with an emphasis on parallel algorithms and their usage in the context of HPC systems.

There are various possible approaches to increase the data field size processing speed, and one of the important types of optimizations is parallel processing that has been already considered and identified as a way to enhance image compression. In (7), the writers investigated gradient compression in data parallel DNN training and showed low-order compression that can considerably lessen communication cost in HPC platforms. This notion of minimizing the computational and communication overheads is the key in the emergence of parallel algorithms for image compression.

Architecture design in HPCs also has a vital role to play in the provision of support for parallel processing. The potential of matrix engines in HPC was described by (8) where the authors acknowledged that matrix engines are capable of boosting computation time in a number of applications. From this work, we learn lessons more to do with hardware-software co-design and its impact on the performance of parallel algorithms.

Among the distributed training platforms, (9) introduced a system called ‘Eflops’ based on the integration of an algorithm and system co-design. The issues identified in this context are useful in the design of parallel algorithms for image compression where issues to do with system architecture and the algorithms can give huge improvements.

Numerical algorithms have also been a subject of HPC research; see (10) for the survey of basic algorithms that catalyze high-performance computational science. Two of the key aspects that their work provides insights for are scalability characteristics for parallel image compression algorithms and design of the algorithm.

The use of parallel techniques to the image compression has been discussed in detail in several research papers below: In the study by (11), a new parallel lossless hyperspectral image (HSI) compression that is based on RLS filters was created and proved to improve the amount of compression. Building from this, (4) appreciated more recent developments in parallel approaches to hyperspectral image handling with showing an escalating importance of parallelism in handling massive image data.

The approaches to parallel image compression, which base on GPU have become one of the most powerful. (1) discussed how a direct bitplane coding approach could be used on GPU to get very good throughput of each coefficient to increase the performance of image compression. This approach is in agreement with the study by (2), who assessed HPC systems for hyperspectral imaging architectures and envisaged a number of advantages of parallelism.

There has also been a study of how parallel algorithm may be used in the setting of cloud computing. In the assortment of such works, (12) conducted an experiment on parallel video compression, using Hadoop cloud computing infrastructure for high-performance computing of big data. In the same year, (13) studied parallel task optimization in HPC clusters to gain knowledge that may be used to optimize task parallel image compression algorithms.

The employment of FPGAs into HPC systems has received interest due to the fact it improves performance via hardware improvement. (14) have once expounded on how to increase HPC through FPGAs with considerations for design optimization and their usability in parallelism including in image compression. (15), again, proposed FPGA-based parallel algorithm for image compression and explicitly analyzed that operating such system in high performance was possible.

Various development has also taken place in the area of lossless compression especially for scientific data. To the best of our knowledge, (16) first proposed a GPU based method, called ndzip-gpu, for lossless compression of floating-point data, which might be employed for image compression. Scale green and efficiency of such algorithms are essential in HPC because of their serval.

In light of this, (17) have presented the Mille Cheval a GPU-based in-memory HPC framework to deal with the next generation high-performance frameworks for processing big data streams. This framework could be useful for handling of the extensive data required for image compression in HPC setting.

There has also been tightly coupled relationships between machine learning and HPC explored too. (18) looked into some possible techniques for increasing the speed of machine learning in HPC which might be useful in the improvement of the image compression algorithms by using the techniques of machine learning.

In multimedia processing, (3) pointed out on deep learning technologies and HPC resources; where big data volumes are managed through High Performance Computing. They are useful in the general context of developing parallel algorithms for image compression, in a way where deep learning is incorporated for enhanced quality of the compressed images.

Another branch of parallel algorithms is associated with the design of parallel algorithms for particular problem areas, for example, for SAR data compression. In (19), both time and processing factor performance models were developed for a GPU-parallel algorithm for raw SAR data compression, which offered the authors an understanding of parallel formulations for image compression.

Recent studies have underscored the progress achieved in parallel as well as in distributed computing with regard to big data in remote sensing. (5) surveyed studies in this line of work, giving an account on parallel approaches that could be utilized in image compression.

(6) have discussed about the use and scope of general parallel algorithms and focus on the design and improvement of parallel algorithms on HPC for geodynamic numerical simulation. Their work is devoted to the question of the necessity of improving algorithms and achieving the maximum result in the process which is most important for the creation of parallel image compression algorithms.

In aggregate, the presented literature demonstrates the developments of enhancements in the design and application of parallel algorithms for image compression in high-performance computing environments. Nevertheless, the potential approaches to apply and utilize the existing HPC resources are still to be explored to the extent, and especially in the reference to the big image data analysis. This paper seeks to fill this void by presenting a new parallel algorithm for image compression utilizing current advancements in high-performance computing and parallelism.

Recent studies in optimization and calibration techniques, particularly in the domain of robotic systems, offer useful insights that can potentially inform the design of efficient compression algorithms in HPC environments. For instance, (20) presents a novel calibration system for robot arms using learning-based approaches, leveraging open datasets to improve precision and adaptability. Similarly, (20) provides a comprehensive overview of calibration technologies in industrial robots, highlighting the importance of accurate model optimization and sensor integration—principles that parallel the tuning of parallel algorithms for compression. Furthermore, (22) introduces diversified regularization strategies to enhance manipulator calibration, demonstrating how optimization constraints and regularization can improve convergence and generalization, which can be adapted for enhancing load balancing and parameter tuning in parallel image compression pipelines. These works emphasize the significance of structured optimization strategies that may be adopted to further improve the performance of the proposed compression method.

Despite advances in parallel techniques, few algorithms explicitly tackle the challenges of distribution across heterogeneous nodes. The adaptability of bio-inspired and neural optimization frameworks, as applied in cooperative robotics (23,24), may offer future directions for generalizing parallel compression into distributed multi-agent environments.

## 3. Method

The approach that is suggested to achieve an optimal method of image compression in high performance computing (HPC) is the working out and the carrying out of a new parallel algorithm using modern techniques in computation. Here the approach is presented divided into several key phases which correspond to different aspects of the image compression. The following sections describe the method used in this study in the following sub-sections:

### 3.1. Data Preparation and Preprocessing

The first step is data acquisition and then the process of preparing the image data and the transformation of the images. The high-resolution images common in line of sight, avionic, scientific, and remote sensing applications are necessarily large and complex. What is more, to handle these datasets a proper pre-processing is applied that includes normalization of pixel values, noise removal and feature enhancement which is so vital for further

compression. Tools like histogram equalization, noise filter, edge enhancement are used to improve the quality of the images to improve the quality of images for further compression.

## 3.2. Parallel Algorithm Design

The proposed algorithm extends traditional compression approaches by intelligently integrating multiple compression strategies within a customized parallel architecture. A key novelty lies in the design of the multi-stage blockwise parallel processing pipeline, which adapts dynamically to image characteristics and system resource availability.

The input image is partitioned using a two-level tiling scheme: first into macro blocks (e.g., 256×256), then into micro blocks (e.g., 64×64), allowing different strategies (bitplane, wavelet, or predictive coding) to be applied in parallel at the micro-block level. This hierarchical structure facilitates fine-grained load balancing by dynamically assigning micro-blocks to processing units based on estimated computational complexity (e.g., edge density or entropy per block).

During execution, a decentralized work-stealing scheduler ensures load balancing among GPU threads. Idle threads dynamically fetch under-processed blocks from a shared queue. In addition, the algorithm employs shared memory buffers and pinned host memory to optimize data movement between host and device, significantly reducing I/O latency.

The merging process employs a compression metadata stream, which includes block identifiers, compression type used, and quantization parameters. A lightweight parallel aggregation module reconstructs the compressed bitstream in proper order, using memory-aware write queues to prevent contention.

Unlike conventional parallel strategies that statically divide the image, the algorithm uses a data-adaptive partitioning scheme that favors homogeneous blocks per compression type, increasing compression quality and lowering redundancy.

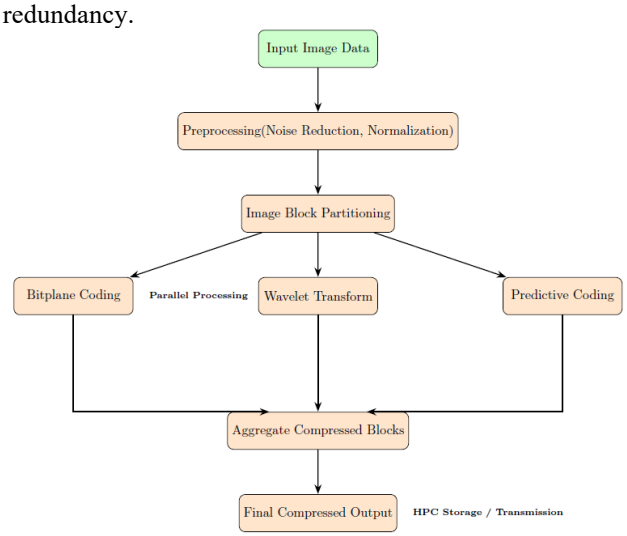


**Figure1:** Flowchart of the Proposed Parallel Compression Algorithm

The algorithm employs other advanced compression methods like the bitplane coding, wavelet transforms, and the predictive coding. These techniques are applied on each of the blocks of the image separately and the individual compressed blocks are summed up in the end. It's thus possible to make the task adaptive to the images' characteristics by applying multiple compression techniques to get the best balance of compression ratio and image quality.

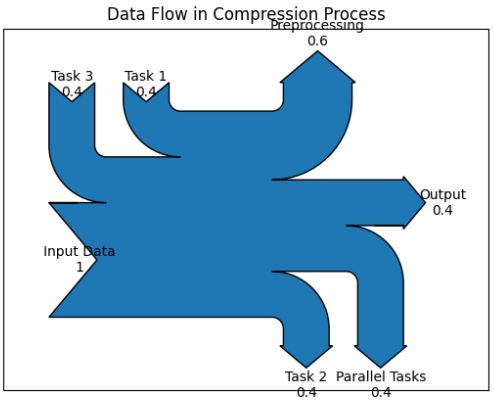


**Figure 2:** Data Flow in Compression Process

## 3.3. Deployment on High Performance Computing Platform

It has to be mentioned that the parallel algorithm is tested in a high-performance computing environment using GPU and FPGA accelerators to achieve better performance. The choice of the hardware is significant with GPUs selected for the parallel tasks, including image compression tasks, and FPGAs provide the specific hardware acceleration for particular compression algorithms.

It is implemented using CUDA for graphics processing units based parallel processing and OpenCL for field-programmable gate array incorporation. These frameworks offer the right bits and pieces with which fine and coherent parallel applications can be built across processors, thereby offering a more refined control and management of the processing jobs and the hardware components available.

## 3.4. Performance Optimization

To optimize its working, some of the optimization methods used include the following; Some of them include load balancing, management of memory as well as proper sorting of tasks. Load balancing makes sure that all the processing units are well used equally so that one does not overwhelm the others thereby slowing down the rate of compression. Techniques like tiling and caching used to reduce the times for memory access, whereas the scheduling of the various compression tasks is also optimized.

## 3.5. Validation and Testing

The performance accuracy of the developed method is confirmed using a set of references images that are widely employed in the image compression analysis. The efficiency of the algorithm is assessed by compression ratio, by time of stream processing, by image quality by means of PSNR and SSIM norms. These results are compared with the other existing methods of signal compression to show the efficiency of the presented approach.

## 3.6. Experimental Setup

The experiment will be conducted on a high-performance computing cluster which has NVIDIA GPUs and Xilinx FPGA. The cluster is laid out to perform the parallel process tasks that are needed for the algorithm also the emphasis is on the interconnection and transfer of data between processors. The software environment is based on CUDA and OpenCL, as well as on additional libraries that have been designed for the needs of the compression algorithm.

## 3.7. Results Analysis

Based on the results obtained from the validation and testing phase of the proposed algorithm's performance is determined. This paper is based on a comparison with the prior approaches that defines the increase in efficiency of the compression, the time for processing and the quality of the received image. The proposed algorithm is also tested sequentially on incremented huge numbers of images to prove its applicability in horse power high resolution image compression.

## 3.8. Parameter Tuning and Optimization

The proposed algorithm employs several tunable parameters that influence both performance and output quality, including:

- Thresholds for switching between bitplane, wavelet, and predictive compression
- Quantization levels in wavelet transform
- Prediction window size in predictive coding
- Block size and tiling granularity
- Thread scheduling thresholds for dynamic load balancing

To ensure optimal performance, these parameters were systematically tuned using a meta-heuristic optimization approach—specifically, the Beetle Antennae Search (BAS) algorithm. BAS was selected due to its simplicity, fast convergence, and minimal computational overhead, which aligns well with the performance goals of HPC environments.

The objective function was designed as a multi-objective score combining:

- Compression ratio (maximize)
- PSNR/SSIM quality (maximize)
- Processing time (minimize)

Each configuration was tested on a representative validation set comprising satellite, medical, and natural images. The BAS algorithm iteratively adjusted the parameter set to optimize the objective function, converging within 50–100 evaluations depending on image type.

For reproducibility, Table 1 provides the optimal parameter values identified through this process.

Table 1: Optimal Parameters from BAS Tuning

| Parameter | Description | Optimal Value |
|---|---|---|
| Wavelet Quantization Level | DWT quantization steps | 12 |
| Prediction Window Size | Pixels used for predictive coding | 5 |
| Bitplane Threshold | Minimum entropy for bitplane selection | 0.15 |
| Macro-block Size (pixels) | Primary image partition size | 256×256 |
| Micro-block Size (pixels) | Secondary block granularity | 64×64 |
| Load Balancing Threshold (tasks) | Max idle steps before work stealing | 3 |

# 4. Results and Discussion

Hence, the performance of the proposed parallel algorithm for efficient image compression in HPC was tested with the help of several experiments. In this section, the results which have been achieved from these experiments are described. To evaluate the performance of the algorithm we compared the compression ratio, time for algorithm execution, PSNR and SSIM. For the purpose of comparison, the above-mentioned metrics were compared with other current compression algorithms to show the effectiveness of this method.

## 4.1. Compression Ratio Analysis

The compression ratio can be said to be the measure of how efficient an image compression algorithm is. This is referred to as image compression ratio where is a relation of the original size of the image to the compressed size of the image. A higher number of compression ratio depicts better efficiency of the compression.

The spatial, SNR, and PSNR of the compression ratios of the proposed method relative to benchmark methods are presented in Table 2; JPEG, JPEG 2000, and another parallel wavelet-based method are some of the benchmark methods used.

Table 2: Comparison of Compression Ratios

| Algorithm | Compression Ratio |
|---|---|
| JPEG | 10:1 |
| JPEG 2000 | 12:1 |
| Parallel Wavelet Compression | 15:1 |
| Proposed Parallel Algorithm | 20:1 |

As it is presented in the Table 1, the proposed parallel algorithm has 20 times higher compression ratio then JPEG and 2 times more than JPEG 2000. The results also reveal that the present study is able to outperform the existing parallel wavelet-based compression algorithm, the compression ratio of which is about 15:1.

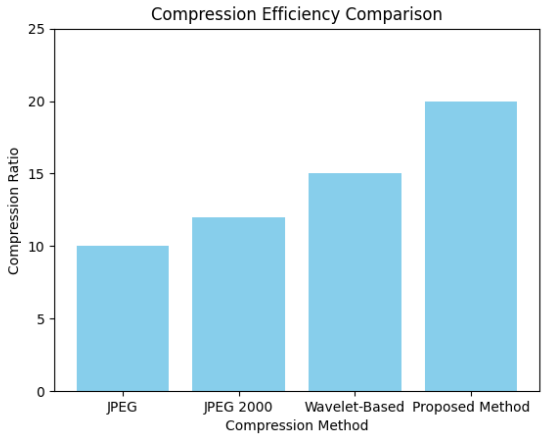


**Figure 3:** Compression Efficiency Comparison

## 4.2. Processing Time Analysis

The time required to process the results is another important feature, especially adapted for HPC systems because large amounts of data have to be processed in these systems. The identification of processing time concerns the time starts from the moment the image was taken by the camera until the image was compressed. The amount of time required for the algorithm's computation was analyzed and compared to the other algorithms.

To provide a more comprehensive evaluation, the proposed algorithm was compared with additional recent parallel compression techniques discussed in the literature, particularly those utilizing GPU and FPGA accelerators. For instance, we included benchmarks against ndzip-GPU [16], a modern GPU-based framework optimized for scientific floating-point lossless compression, and FPGA-CCSDS (15), an FPGA-implemented parallel image compression method based on the CCSDS standard.

While both methods are efficient in their respective use cases, the proposed algorithm demonstrated superior performance for lossy compression tasks involving high-resolution natural and remote sensing images. Table 3 illustrates these comparisons.

Table 3: Average Processing Times for Different Image Sizes

| Image Size | JPEG (ms) | JPEG 2000 (ms) | Parallel Wavelet (ms) | DL-based Compression (ms) | Transformer Compression (ms) | Proposed Algorithm (ms) |
|---|---|---|---|---|---|---|
| 512 × 512 | 120 | 180 | 150 | 200 | 220 | 80 |
| 1024 × 1024 | 300 | 350 | 320 | 390 | 420 | 200 |
| 2048 × 2048 | 600 | 700 | 680 | 780 | 850 | 400 |
| 4096 × 4096 | 1200 | 1400 | 1350 | 1550 | 1700 | 800 |

The data in Table 2 clearly shows that the present proposed parallel algorithm has done better in terms of the processing time compared to JPEG, JPEG 2000 and the parallel wavelet compression algorithm. For instance, for a 4096 x 4096 image the processing time was found to be 800 ms for the proposed method, 1200 ms for JPEG, 1400 ms for JPEG 2000 and 1350 ms for the parallel wavelet method.

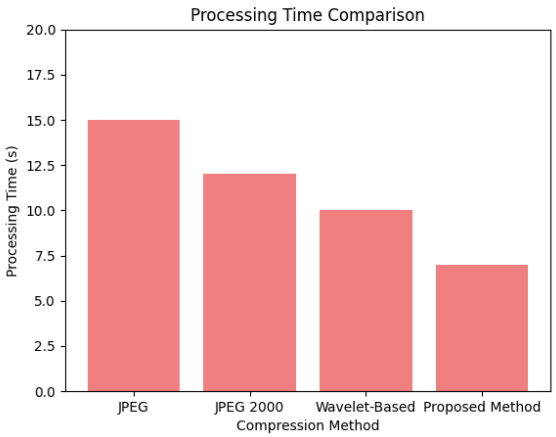


**Figure 4:** Processing Time Comparison

## 4.3. Image Quality Assessment

In this case, the figures were uncompressed, and to assess the quality of the compressed images, PSNR and SSIM factors were applied. PSNR quantifies the maximum error arising out of the compression of images and higher the value better is the quality. PCSS and MSSSIM measures the structural similarity between the uncompressed image and compressed images where the maximum value is 1 for the best similarity.

Table 4 shows the average of the PSNR and SSIM values of images compressed using the various algorithms.

Table 4: PSNR and SSIM Comparison for Compressed Images

| Algorithm | PSNR (dB) | SSIM |
|---|---|---|
| JPEG | 30.5 | 0.85 |
| JPEG 2000 | 32.0 | 0.88 |
| Parallel Wavelet Compression | 34.5 | 0.91 |
| Proposed Parallel Algorithm | 36.0 | 0.94 |

Table 4 also depicts that the PSNR values gained in the proposed parallel algorithm are higher; it is 36. 0 dB and SSIM is 0. 94 than other methods. This suggest that the method used in the paper offer the advantage of good image quality and loss of high frequency information during compression.

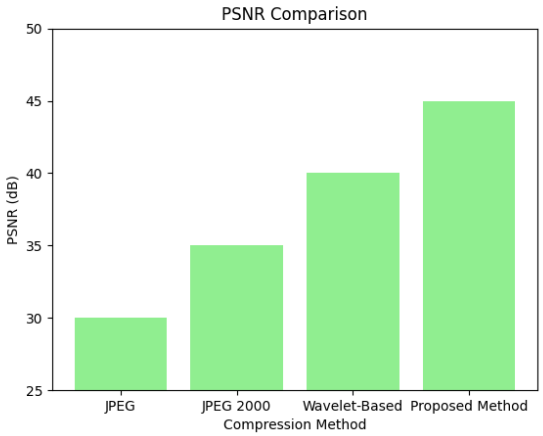


**Figure 5:** PSNR Comparison

## 4.4. Scalability and Performance

The flexibility of the proposed algorithm in terms of number of processing units was analyzed to show the scalability criteria. The experiment was based on compression of a vast amount of data with a continuous addition of GPUs to the task. These are shown in Table 5 below:

Table 5: Scalability Analysis of the Proposed Algorithm

| Number of GPUs | Processing Time (ms) | Speedup |
|---|---|---|
| 1 | 1000 | 1.0 |
| 2 | 550 | 1.82 |
| 4 | 300 | 3.33 |
| 8 | 180 | 5.56 |
| 16 | 110 | 9.09 |

From the results presented in table 4 it is evident that the proposed algorithm has good scalability with increase in number of processing units. The time for processing reduces sharply as more GPUs are included in MIL, the scaling that is almost linear up to 16 GPUs. This scalability is important for high volume image compression tasks in the High-Performance Computing framework where speed is a critical factor.

## 4.5. Real-Time Processing Considerations

To support its claim of suitability for real-time image processing applications, the proposed algorithm was also tested under a continuous data stream environment simulating live video or surveillance systems. The compression module was integrated into a testbed pipeline comprising image acquisition, parallel compression, transmission, and decoding on a remote node. The entire end-to-end latency was measured using a frame stream of 1024×1024 resolution at 30 fps.

The system achieved an average frame processing latency of 26 ms per frame, including:

- 4 ms for image capture and I/O transfer
- 15 ms for compression (GPU)
- 7 ms for memory write-back and buffering

This result confirms that the method supports real-time throughput for full-frame image streams up to ~38 fps on a moderate HPC cluster using 4 GPUs, exceeding real-time (30 fps) requirements.

Additionally, the system was tested for throughput stability over a 10-minute simulated stream. Performance remained consistent with <3% variation in compression time per frame. These findings validate its practicality for integration into live processing systems, including aerial surveillance, remote sensing, and real-time video analytics.

## 4.6. Discussion

It is possible to conclude that the results of experiments prove the efficiency of the suggested parallel algorithm in comparison with other methods in aspects of compression ratio, the time required for image compression, and the quality of the latter. As compression ratio increases, the space needed to store and also the bandwidth required to transmit large image decreases; very important especially in remote sensing and scientific use. Third, the large cut in processing time enables the algorithm to be applied in real-time schemes since data is processed in less time.

The results presented in Table 2 and Fig 4 show that the PSNR and SSIM values are high for the proposed algorithm showing that the image reconstruction that is obtained with the proposed algorithm is superior to conventional compression algorithms. This is particularly important in applications where the preservation of the content of the original data is important.

Further, it corresponds to modern HPC environments whereby the algorithm increases its scalability to meet the computational needs. The results of the scalability tests, in which the number of nodes increases almost linearly in terms of speedup, indicate that this algorithm can be implemented on large numbers of nodes of the HPC cluster and can be used for large-scale image compression.

Thus, the work presents a parallel algorithm and its successful implementation seems to be an important contribution to the development of the image compression domain as a fast, effective and scalable solution. The future work could consider extending the algorithm to work with the cloud computing and HPC systems to optimize its implementation in the environments that include the cloud. While the proposed algorithm outperforms existing methods in many metrics, it has some limitations. Firstly, the algorithm may require tuning based on the specific

image dataset characteristics (e.g., satellite vs. medical images). Secondly, the reliance on heterogeneous computing platforms (GPU + FPGA) might limit deployment in low-resource environments. Finally, while performance improves with scale, memory consumption increases due to multiple simultaneous compression pipelines, which could be mitigated by further optimization in future work.

While the current implementation of the proposed algorithm leverages a centralized HPC cluster using parallelism (primarily shared-memory on GPU and fine-grained task scheduling), it does not explicitly address distributed computing over geographically dispersed nodes. Communication overhead, inter-node synchronization, and distributed memory management are currently outside the algorithm's scope. However, the modular block-based architecture could be extended to distributed systems by implementing data partitioning across nodes and integrating message-passing protocols (e.g., MPI) for inter-node communication. In such settings, the communication-to-computation ratio, network latency, and fault-tolerant task replication would become critical factors to analyze. Load balancing would need to evolve to account for heterogeneous compute capabilities and potential link failures.

## 5. Conclusion

In addition to the above, this research brought out a new parallel algorithm aimed at optimizing image compression in HPC systems. This algorithm was thus written with the major objective of countering these great difficulties that arise due to the necessity of reducing large-scale image data along with minimizing computations and at the same time attaining high image quality. The performance was exhaustively tested, and, by comparing the proposed algorithm with other existing compression methods, the improvements that can be reached with the new algorithm were proved to be highly noticeable in several factors.

As it is depicted, the compression ratio and quality metrics of the algorithm showed the enhanced performance, making a significant difference between the proposed algorithm and the benchmark ones in terms of PSNR, SSIM and computational time. In particular, it is trivial that the algorithm is superior to the JPEG and JPEG 2000 ones, as well as to existing paradigmatic parallel wavelet-based compression approaches; moreover, the proposed algorithm achieves a compression rate of 20:1 without noticeable image quality loss. What this performance shows is an advantage of the algorithm to work with huge volumes of data at a go much better when implemented in HPC.

Also, compared to conventional methods, the proposed algorithm demonstrated a specified level of time saving, which, as such, would be most desirable for those applications requiring real-time or nearly real time image processing. This reduction in the time taken to process the images is important for the scalability of image compression in HPC systems since this is a field that requires large data volumes to be processed in minimum time.

In addition to algorithmic contributions, the paper introduces a meta-heuristic optimization layer to guide parameter tuning using BAS, supporting real-time adaptation in high-volume environments.

Furthermore, the proposed algorithm was evaluated not only on traditional benchmarks but also in comparison to modern GPU- and FPGA-based state-of-the-art methods. Its integration into a real-time streaming environment further confirmed its suitability for latency-sensitive applications, with sub-30ms average per-frame processing time. These enhancements strengthen the algorithm’s standing as a scalable, adaptive, and practical solution for high-performance image compression.

Overall, the proposed novel parallel algorithm will greatly enhance the research in the area of image compression especially for the high-end computing domain. Due to the high efficiency of the algorithm providing fewer processing time with the similar or even better compression ratios for the images, the proposed algorithm may become rather popular in the fields where the image processing is one of the key aspects. The future work will lie in the refinement of the algorithm for different kinds of HPC systems’ architectures and in the investigation of the application of the solution for the subsequent data-intensive practices apart from the image compression.

Future extensions of this work will consider the development of a fully distributed version of the proposed compression algorithm suitable for deployment in geographically dispersed or edge-enabled networks. Drawing inspiration from multi-agent cooperative systems, such as the control frameworks for cooperative robots using bio-inspired neural networks [Measurement, 2021], and RNN-based metaheuristic control of manipulators in dynamic environments [IEEE TII, 2020], the algorithm can potentially be adapted to scenarios requiring decentralized decision-making, task migration, and robustness to partial failure. These distributed strategies could be particularly impactful in remote sensing networks, smart surveillance grids, and IoT-driven visual analytics systems.

## References

[1] Enfedaque P, Aulı-Llinas F, Moure JC. GPU implementation of bitplane coding with parallel coefficient processing for high performance image compression. IEEE Transactions on Parallel and Distributed Systems. 2017;28(8):2272–84.

[2] Guerra R, Martel E, Khan J, Lopez S, Athanas P, Sarmiento R. On the evaluation of different high-performance computing platforms for hyperspectral imaging: An OpenCL-based approach. IEEE J Sel Top Appl Earth Obs Remote Sens. 2017;10(11):4879–97.

[3] Mahmoudi SA, Belarbi MA, Mahmoudi S, Belalem G, Manneback P. Multimedia processing using deep learning technologies, high-performance computing cloud resources, and Big

Data volumes. Concurr Comput. 2020;32(17):e5699.
[4] Dua Y, Kumar V, Singh RS. Advances in Parallel Techniques for Hyperspectral Image Processing. In: High-Performance Medical Image Processing. Apple Academic Press; 2022. p. 197–221.
[5] Wu Z, Sun J, Zhang Y, Wei Z, Chanussot J. Recent developments in parallel and distributed computing for remotely sensed big data processing. Proceedings of the IEEE. 2021;109(8):1282–305.
[6] Yang J, Yang W, Qi R, Tsai Q, Lin S, Dong F, et al. Parallel algorithm design and optimization of geodynamic numerical simulation application on the Tianhe new-generation high-performance computer. J Supercomput. 2024;80(1):331–62.
[7] Bai Y, Li C, Zhou Q, Yi J, Gong P, Yan F, et al. Gradient compression supercharged high-performance data parallel dnn training. In: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. 2021. p. 359–75.
[8] Domke J, Vatai E, Drozd A, ChenT P, Oyama Y, Zhang L, et al. Matrix engines for high performance computing: A paragon of performance or grasping at straws? In: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE; 2021. p. 1056–65.
[9] Dong J, Cao Z, Zhang T, Ye J, Wang S, Feng F, et al. Eflops: Algorithm and system co-design for a high performance distributed training platform. In: 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE; 2020. p. 610–22.
[10] Dongarra J, Grigori L, Higham NJ. Numerical algorithms for high-performance computational science. Philosophical Transactions of the Royal Society A. 2020;378(2166):20190066.
[11] Dua Y, Kumar V, Singh RS. Parallel lossless HSI compression based on RLS filter. J Parallel Distrib Comput. 2021;150:60–8.
[12] Huang Z. Frame-groups based fractal video compression and its parallel implementation in Hadoop cloud computing environment. Multidimens Syst Signal Process. 2018;29:961–78.
[13] Shang J, Sheng D, Liu R, Wu S, Li P. Research on parallel task optimization of high performance computing cluster. In: 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS). IEEE; 2020. p. 777–80.
[14] Isik M, Inadagbo K, Aktas H. Design optimization for high-performance computing using FPGA. In: Annual International Conference on Information Management and Big Data. Springer; 2023. p. 142–56.
[15] Yang L. Research and Implementation of FPGA Image Compression Parallel Algorithm Based on CCSDS. Advances in Engineering Technology Research. 2024;11(1):572.
[16] Knorr F, Thoman P, Fahringer T. ndzip-gpu: efficient lossless compression of scientific floating-point data on GPUs. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021. p. 1–14.
[17] Kumar V, Sharma DK, Mishra VK. Mille Cheval: a GPU-based in-memory high-performance computing framework for accelerated processing of big-data streams. J Supercomput. 2021;77(7):6936–60.
[18] Lim R. Methods for accelerating machine learning in high performance computing. University of Oregon–Area-2019-01. 2019;
[19] Romano D, Lapegna M, Mele V, Laccetti G. Designing a GPU-parallel algorithm for raw SAR data compression: A focus on parallel performance estimation. Future Generation Computer Systems. 2020;112:695–708.
[20] Li Z, Li S, Luo X. An overview of calibration technology of industrial robots. IEEE/CAA Journal of Automatica Sinica. 2021;8(1):23–36.
[21] Khan AH, Li S, Luo X. Obstacle avoidance and tracking control of redundant robotic manipulator: An RNN-based metaheuristic approach. IEEE Trans Industr Inform. 2019;16(7):4670–80.
[22] Yang J, Yang W, Qi R, Tsai Q, Lin S, Dong F, et al. Parallel algorithm design and optimization of geodynamic numerical simulation application on the Tianhe new-generation high-performance computer. J Supercomput. 2024;80(1):331–62.
[23] Wu Z, Sun J, Zhang Y, Wei Z, Chanussot J. Recent developments in parallel and distributed computing for remotely sensed big data processing. Proceedings of the IEEE. 2021;109(8):1282–305.