Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9810896, 1 page
https://doi.org/10.1155/2023/9810896

Retraction

Retracted: Heterogeneous Cluster Application Communication Optimization and Computer Big Data Management

Wireless Communications and Mobile Computing

Received 18 July 2023; Accepted 18 July 2023; Published 19 July 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] H. Wang, "Heterogeneous Cluster Application Communication Optimization and Computer Big Data Management," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 1106003, 7 pages, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1106003, 7 pages
https://doi.org/10.1155/2022/1106003

Research Article

Heterogeneous Cluster Application Communication Optimization and Computer Big Data Management

Hongyan Wang

Computing Center of Anshan Normal University, Anshan, Liaoning 114007, China

Correspondence should be addressed to Hongyan Wang; 201903322@stu.ncwu.edu.cn

Received 17 June 2022; Revised 19 July 2022; Accepted 26 July 2022; Published 4 August 2022

Academic Editor: Balakrishnan Nagaraj

In order to adapt to the constantly updated heterogeneous hardware and super-large-scale parallel computing environment and solve the problems of low programming level and difficult development, modification, and debugging of commonly used solutions, the author proposes a method for application communication optimization and computer big data management based on heterogeneous clusters. A new language mechanism is introduced to describe the multidimensional rule structure, arrangement, and communication mode of data and threads; and a software migration and optimization method between different types of heterogeneous systems based on the new language mechanism is proposed. And take direct method turbulence simulation as an example. Experimental results show that based on the Parray mechanism, it took only one week to complete the rapid migration of turbulence simulation applications on the Tianhe 1A system, and it was successfully run on a scale of 8192 cubic meters. *Conclusion*. The method realizes communication optimization and fast porting in different heterogeneous systems.

1. Introduction

In recent years, deep training has been successfully implemented in various skills such as drawing and natural language processing, as shown in Figure 1. The time is very long, usually days or even weeks, and in order to improve the positioning of the deep training standard, the measurement of data is too numerous, the time required for training also increases, and the computing power of the computer is limited and cannot be satisfied on demand. In order to improve the efficiency of system training, distributed training has been carried out on the cluster in recent years, and the training process on the original single machine is distributed to multiple machines for parallel execution, which improves the processing speed of data samples and greatly shortens the training time, for example, recently, HUAWEI CLOUD ModelArts used 16 nodes and 8 v100 GPUs per node to train ResNet-50 in a cluster distributed manner, and it only took 10 minutes and 28 seconds to converge on the ImageNet dataset. Distributed deep learning is a necessary means to cope with the increasing scale of data and models, becoming a key issue in both academia and industry [1].

When performing distributed training in a heterogeneous cluster, due to the large differences in computing and network performance of different machines, the iteration time of different worker nodes under the same workload will also vary greatly. When using the BSP algorithm, since each iteration needs to wait for all worker nodes to complete, the performance of distributed training is limited by the slowest worker node [2]. When using the ASP algorithm, each worker node updates parameters independently, and after completing some iterations, it can start the next iteration without waiting for other worker nodes, but this will make each worker node train based on different parameters, especially when some worker nodes are significantly slower than others, the slow worker nodes have completed multiple iterations and updated parameters during one iteration, while the slow worker nodes still calculate based on the old parameters before, however, the gradient update parameters obtained by training with outdated parameters will cause the parameters to deviate from the optimal solution, resulting in incorrect convergence, thereby slowing down the convergence speed [3].

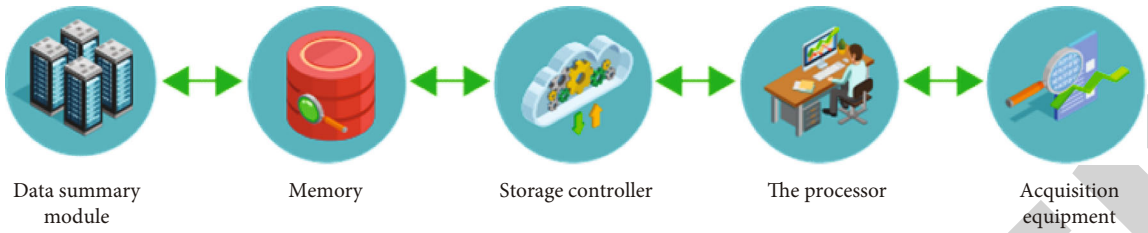

Figure 1: Flowchart of computer big data management.

Table 1: Features of existing cluster programming languages.

Language	Accomplish		Address space			Control	
	Language	Library	Global	Local	Cluster	Multicore	Many core
Chapel	√		√		√	√	
CAF	√		√		√		
HPF	√		√		√		
HTA		√	√		√	√	
MPI/PVM		√		√	√		
Titanium	√		√		√		
UPC	√		√		√		
X10	√		√		√	√	
ZPL	√		√		√		

2. Literature Review

Wang et al. created a software library capable of supporting C++ applications, using a hybrid implementation of the MIC coprocessor, communicating with SCIF for data transfer and synchronization. Although COI and SCIF have their own research work and programming guidance, so far, no researchers have proposed a programming method that uses COI and SCIF mixed. Many core algorithmic programs and real-world applications have been studied on MIC coprocessors [4]. Jayakumar et al. summarized the key techniques for obtaining high performance of MIC coprocessing [5]. Xu et al. ported an existing scientific computing application and a number of microkernels to a single MIC coprocessor [6]. Li et al. developed a SIMD molecular dynamics application on a single MIC coprocessor [7]. Goldenberg et al. accelerated large-scale sparse linear system iterative algorithm PQMRCGSTAB, image and video compression IDCT algorithm, molecular dynamics simulation application, etc. on MIC coprocessing and studied the automatic conversion method and optimization of offload code from OpenACC to Intel offload mode method [8].

In terms of programming framework research, Bachiller et al. proposed the Uintah software computing framework and solved the interaction problem of solving multiple fluid structures on an adaptive structured grid [9]. For Uintah, the problem of solving complex multiscale and multiphysics fields is studied, by integrating various simulation components, users can describe the dependencies between components through DAG diagrams, automatically generate parallel code and handle load balancing. The fluid structure joint interaction simulation (Uintah AMR MPMICE) is carried out on Stampede accelerated by coprocessing, by using peer-to-peer MPI communication, the MPI process is run separately on the host CPU and the coprocessor, the device is regarded as an independent node, and it performs cooperative calculation through MPI communication. The MPI process on the device develops CPU multicore or coprocessor many-core parallelism through Pthreads multithreading. The host side of each node in this study starts an MPI process, each MPI process develops sixteen Openmp threads, and the coprocessor side is Two MPI processes, and each MPI process develops sixty OpenMp threads. The application scales up to sixteen nodes, each with a MIC coprocessor. However, Uintah has a load imbalance problem on heterogeneous systems and even affects computing performance in nearly 60% of cases.

Taking the direct turbulence simulation method as an example, the author discusses the new software that should be used in this calculation from the perspectives of software compatibility, programming process, algorithm design and implementation, and software support and provides experience in various applications of the exchange software [10].

3. Research Methods

Existing compound sentences are often used in native language groups. Table 1 lists the characteristics of the existing programming paragraphs. Among them, the first row is categorized by how the language is used: (1) the operations of data communication and sharing are realized in the form of libraries, while other operations are represented by traditional serial language elements; (2) the parallel semantics are expressed explicitly or implicitly by language constructs or primitives. The second column categorizes the languages according to the address space visible to each execution

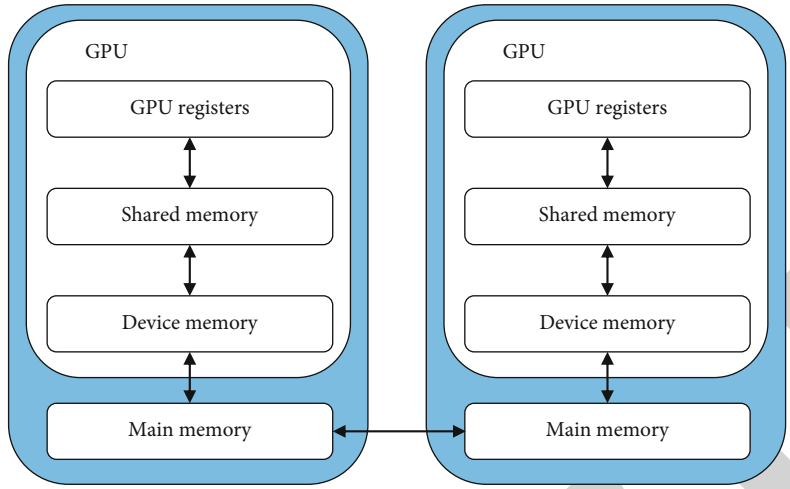

Figure 2: Multilayer storage structure of GPU cluster.

element of the parallel program: in MPI, processes can only access data directly from the local node. Other programming models allow threads to access global data at any node. The third line describes resource management in different languages [11].

The performance of data access, such as the ease of operation, can be divided into bandwidth constraints (such as storage bandwidth and communication bandwidth), data constraints, and external memory. During the counting process, important data can be stored in one large memory, multiple accelerator cards, or other memory. Multicore processors are now perfect for heavy duty applications that seem to have high performance. Computation-intensive tasks suitable for many-core acceleration (such as dense matrix multiplication and LINPACK) can generally achieve satisfactory performance utilization on many-core clusters, and the data storage location has little effect on performance.

However, the storage bandwidth is high, but the data area is good. Designing low-bandwidth communication functions (such as count disparity) requires data directly on the accelerator card for large memory and network connectivity [12, 13]. Using high bandwidth communications such as FFT, data can be stored directly in memory and counted by multiple card accelerators. This category of applications requires the bandwidth of the PCI bus to match the network bandwidth; otherwise, the bottleneck of the entire system will appear at the narrowest bandwidth.

For high-bandwidth communications such as FFTs, the performance of each band is often limited by node network bandwidth, bandwidth of multiple PCI interfaces, accelerator performance, GPU count speed, etc. itself. The speed of data transfer between the network and the PCI bus far exceeds that of the card. Such problems do not ostensibly benefit from many-core acceleration, but the implementation in many-core clustered FFT shows that many-core cluster architecture is beneficial to increase the effective total bandwidth of single-node memory and processing units. Additionally, the card can store more than the CPU cache, which means large operations can be sent to memory at once and run faster, reducing data to the same level as the average [14]. How to make a processor for high-bandwidth applications, optimization of non-homogeneous parallel computing algorithms usually focuses on reducing the number of data transfers from memory to memory to the processor.

The new 3D FFT algorithm for the first GPU cluster divides the Z dimension into N3-sized 3D data $(Z * Y * X)$ by P nodes and N/P $Y - X$ 2D pages by the amount of memory (X is a tight configuration). Each page is sent to the GPU for 2D FFT computation and back into large memory. All nodes need to reassemble the Z dimension at each node, then swap big data like Alltoall for FFT computation on GPU. GPU memory typically ranges from 3 GB to 6 GB, which is larger than the CPU cache, allowing it to receive very large blocks from main memory for a single operation and charge. Allow limited bandwidth of the main PCI bus. The bandwidth limitation of PCI bus between main memory and GPU is compensated. In contrast, when doing file blocking, the CPU needs to access critical memory more than the large cache.

The difficulty with the above scheme is the need to change the size of the matrix divided by each cluster. Heterogeneous clusters have more multitier structures than traditional systems [15]. Taking the GPU cluster as an example, as shown in Figure 2, the data stored in the memory is very important, but including the GPU, the data must be stored in the GPU memory; at the same time, in order to utilize GPU for high-performance computing, it is necessary to consider the shared memory and register structure of GPU and perform targeted programming. Therefore, many groups have more options for large changes of matrices in the FFT than the key memory model for existing groups.

There are several ways to identify changes in cluster distribution in different groups: (1) large communication interfaces like Alltoall can switch locations from nodes; (2) main memory can carry tens of GB of data exchanged in scale; (3) GPU can carry high-speed small matrix switches [16, 17]. Fixing all the benefits of network communication, how memory bandwidth is still less than GPU memory bandwidth, is not the key to algorithm optimization, but how to

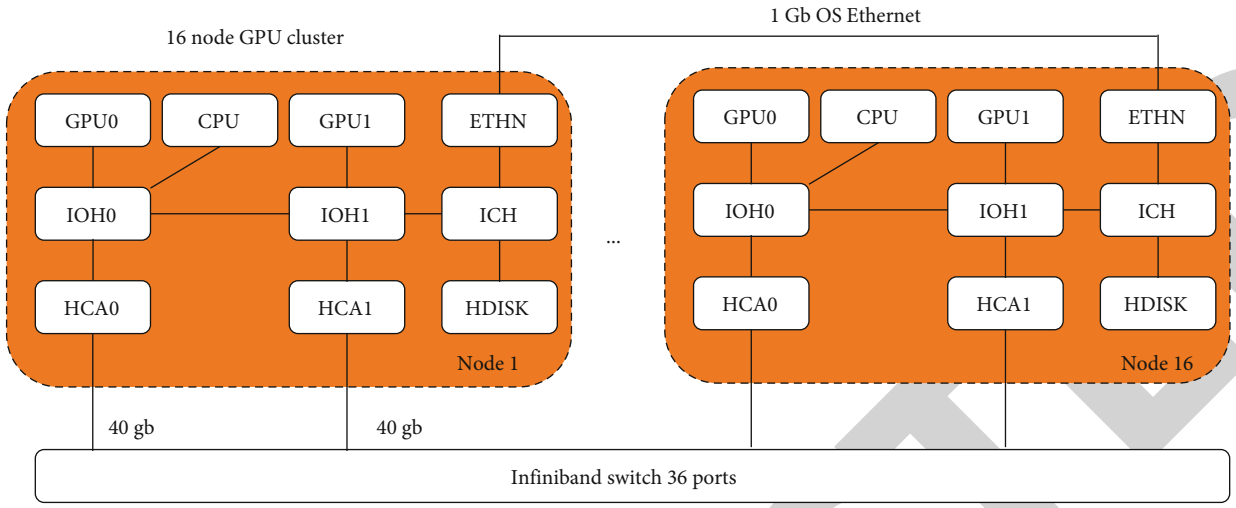

Figure 3: Architecture of PKU McClus.

reduce memory performance. The main innovation is to further divide the data into blocks during the data transmission process and adjust the relative offset of the data blocks, so that the connection between this "offset switch" and the GPU transposition can only be completed for three hours of large, medium, and small. Memoryless transposes in the main move function. Note: different communication systems have different rules for communication granularity (such as the length of data segments). For example, a PCI transfer from critical memory to the GPU requires more than 2 MB of storage to almost peak, while the optimal density for an Infiniband network requires more than 10,000 bytes. Because data sharing reduces the complexity of relationships, the best strategy for saving base memory must take the complexity of relationships into account.

The authors performed validation experiments on a small GPU cluster at PKU McClus to determine the idea of using data movement to FFT groups and adjusting the relative variance of data blocks to minimize memory critical functions [18]. Although the cluster has only 16 nodes, it has a unique architecture, as shown in Figure 3. Each node has two IOHs, each connected to the GPU and Infiniband NIC.

To design dual GPUs and dual Infiniband NICs in a PKU McClus GPU group, we developed detailed procedures for changing the process of distributing data to 2 GPUs per count and using 2 Infiniband NICs to communicate across multiple integrated data channel (directly via IB/veRbs via communication module).

The dimension transformation of the PKUFFT algorithm for this PKU McClus GPU cluster is as follows. Divide the array data into 16 segments according to the length of X, and divide them into 16 nodes. Since they each have 2 GPUs, the data nodes are split into 2 additional parts. Also, the decomposition of the Y length is the same as the decomposition of the X length. First, each GPU performs 128 ($x1$ dimension × $x0$ dimension) 2D FFTs and computes the data in Y- and Z-dimension instructions. The 128 2D FFTs are divided into 32 groups, each group is 4 2D FFTs of 4 096 × 4 096. The loop executes 32 times, sending a set of files to the GPU and 2D each time. Compute the FFT, and pull the result to the Infiniband output, unlike a GPU. In the process of writing from the Infiniband output to the Infiniband output buffer of other nodes, the program transposes the $x3$ dimension and the $y3$ dimension; when the data output by Infiniband is not in the pinned memory of the corresponding GPU, the dimension is adjusted; finally, the transformation is repeated in GPU memory before the 1D FFT of the X dimension after the data is transferred from the memory key to GPU memory. Note: in the subsequent implementation, the GPUDiReCe technology is used, so that the data in the Infiniband output buffer can be directly uploaded to the GPU device memory, thus realizing a complete headless transposition operation.

4. Analysis of Results

4.1. Tianhe 1A Cluster FFT Algorithm and Its Parray Description. Tianhe-1 group and Peking University McClus are two GPU groups, but the models are completely different: each Tianhe-1 group has one GPU and only one communication card. Although the implementation of the PKU McClus FFT algorithm has a similar concept, it needs to provide data sharing for each session and repeat the representation of the communication [19, 20]. The author only describes the implementation of the FFT algorithm for the Tianhe-1 cluster.

We use Parray to represent the pseudocode of the algorithm. For simplicity, the pseudocode only recognizes the array type, not the exchange of real objects. As a benchmark, we employ a single-precision complex float2 (length 8 bytes) to complex (C2C) transform and assume that the data is communicated and back in place after the FFT computation.

In the above virtual code, "2DCUFFT" refers to calling the CUFFT library to perform $N2$ 2D FFT in Y and X dimensions, while "1DBATCHED CUFFT" refers to performing FFT calculation of size N in N dimension Z. The dummy code also gives time estimates for each step. $\beta h2d$ represents the ideal data transmission bandwidth from the main memory to the GPU card (about 5 GB/s). $\beta d2h$ is the

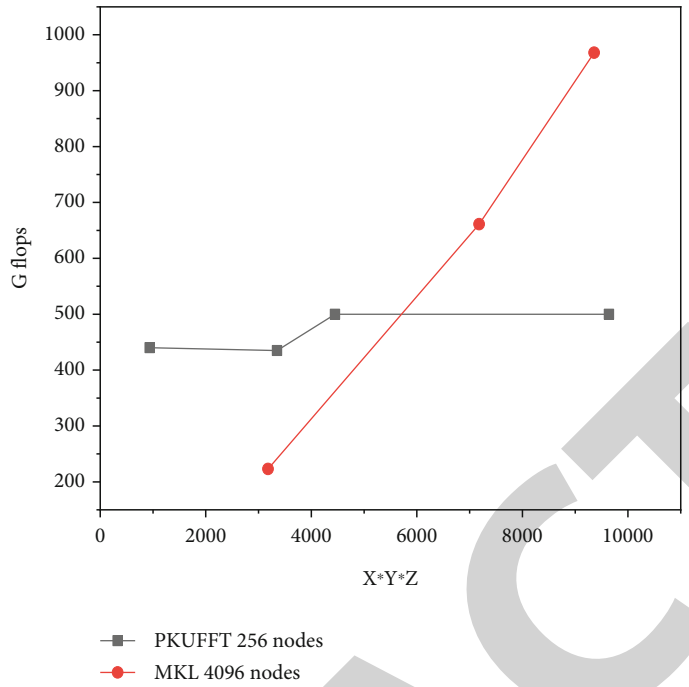

Figure 4: Comparison of Parray 3D FFT code with system performance.

bandwidth of data transmission back to the main memory (about 4 GB/s). $\beta d2d$ is the bandwidth of data transmission in the GPU card (about 100 GB/s). $\beta a2a$ is the average bandwidth of all nodes communicating to each port at the same time (1.2 GB/s for the whole Tianhe machine). And FFT is the single-precision floating-point of reasonable-scale FFT on GPU (the total number of floating-point for a single M-length FFT is 5 Mlog M) calculation speed (about 200Gflops/s). The time-consuming estimation of the three-dimensional FFT with the size of 14336 for the whole 7168 nodes of Tianhe is as follows:

$$\frac{N^3}{P}\cdot\left(2\cdot\frac{8}{\beta_{h2d}}+2\cdot\frac{8}{\beta_{d2h}}+2\cdot\frac{8}{\beta_{a2a}}+2\cdot\frac{8}{\beta_{d2d}}+\frac{5\log N^2}{\phi_{FFT}}+\frac{5\log N}{\phi_{FFT}}\right)\approx 35.8s. \quad (1)$$

We measured the whole machine in Tianhe for 35 s.

The FFT data values used in the turbulence simulations are half of those tested above, split into a forward transform of real to complex (R2C) and a reverse transform of complex to real (C2R). Both real and hard have digit counts, so the exchange rate can be returned to that location without further communication at the end of the line. Note: the R2C and C2R of the CFFT library are slower than C2C, and the model performance varies greatly, so C2C is a necessary measure.

The 3D FFT performance of the GPU group used by Parray was tested on Tianhe 1A and compared with Intel MKL 10.3.1.048 [21]. Figure 4 is the 3D FFT comparison of different scales of the same hard disk (figure PKUFFT is the 3D FFT model of the GPU group used by Tianhe-1 A Parray). It can be seen that the performance of PKUFFT far exceeds that of MKL. Figure 5 shows that PKUFFT has better performance scalability compared to MKL.

4.2. Migration of Direct Method Turbulence Simulation Program. Isotropic direct simulation methods are often used for large Fourier switches. In Tianhe-1A, the measurement of the whole machine can reach 14,336 three-dimensional single-digit real numbers, and the packaging material can reach 11 TB. A turbulent system must have more than a dozen arrays to represent the different components of the system. The competition has shifted from traditional culture to different groups of Tianhe No. 1, and it is necessary to reconstruct the distribution, preparation, and output information according to the characteristics of the heterogeneous groups of buildings.

For the core FFT algorithm used in direct competition, although the FFT algorithm of Tianhe Group 1A is similar to the application algorithm of Peking University McClus, due to changes in product distribution, data preparation and distribution must be reused. Its implementation requires re-coding universally. There are similar problems for porting applications of different heterogeneous groups. The application development based on Parray provides a new mechanism for software migration between different types of heterogeneous systems.

According to the Parray programming interface, agile development and changes of applications in heterogeneous groups can be seen with minimal and fastest code modifications by changing the file types and sizes described in Parray programs. Rapid migration of Parray-based applications typically includes the following steps, please scan the OSID for specific requirements.

(1) Improve the Parray programming interface for new heterogeneous processes, and provide support for new threads and storage formats, this process is done by programmer programmers

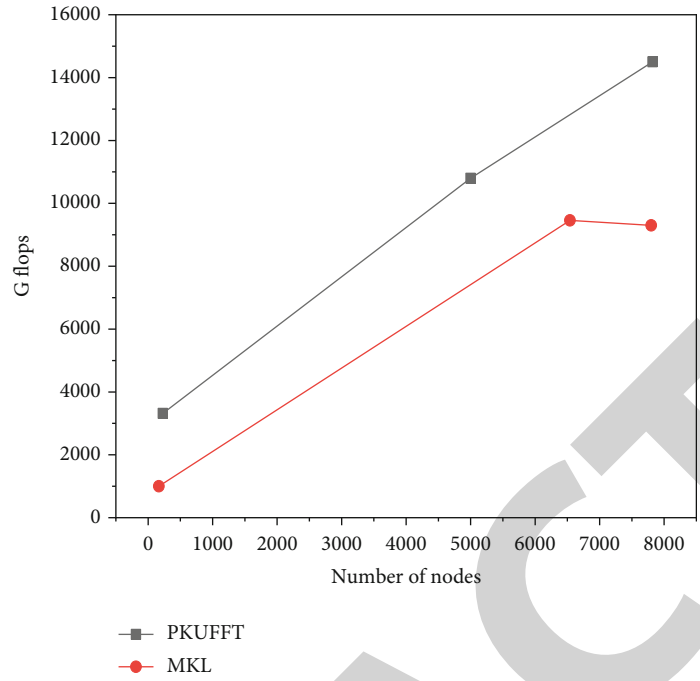

Figure 5: Speedup of Parray 3D FFT code.

(2) For example, for GPU clusters, providing GPU device memory and support for GPU phones through the Parray programming interface, a CUDA strategy must be developed; similarly, new types of support can be quickly introduced to the MIC team

(3) Modifications to applications using Parray are made by the application programmer. For applications not using Parray, use Parray to describe their file distribution and classification, and compare stakeholders; for applications successfully implemented using Parray, the Parray file format and Parray parallelization code need to be modified to accommodate different new demand. During traditional operation, data transfer variables can be looked up by nesting multiple turns, clear description, and application in Parray array mode

(4) Debug new heterogeneous groups for Parray applications. In the process of porting turbulent services, the original code based on MPI is to ensure the accurate operation of heterogeneous groups. The program is not optimized for the same acceleration components and network transfer types. Then, enter the Parray process/thread array mode in the code, and specify the current counting process/thread parallel mode; the Parray data array format is introduced to represent the data array according to the type of application data relationship, and the Parray mixed array format shows the variety of data arrays sex. Allocation is between storage methods and subroutine communication. Simplify the description of data communication methods and optimize transmission. For equations that require composite acceleration, the thread-array type heterogeneous Chinese material is called, further consider the optimal representation of data storage distribution and transmission in heterogeneous computing components and complete the implementation and transplantation of thread computing code in the thread array of heterogeneous computing components

According to the Parray mechanism, we achieved a rapid conversion of the thermal efficiency of the Tianhe-1A system within a week, achieving a measurement of 8192 cubic meters.

5. Conclusion

The authors introduce Parray's communication model to describe the different models, planning, and communication of information and messages. For new heterogeneous processes, Parray can extend the line array type, expand the array support type, and represent new data distribution models; at the same time, by extending and optimizing the performance of subroutine replication, new submissions will be provided. Multithreaded array type integrated telephone can support the transmission of various heterogeneous systems. As an example of using direct turbulence simulation in heterogeneous group transformation, the authors develop an agile method for transforming heterogeneous group programs by changing file types and sizes, based on the Parray programming interface. Minimal and faster than code modification, this process is especially important for the use of heterogeneous groups.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] H. Miyamura, R. G. Bergman, K. N. Raymond, and F. D. Toste, "Heterogeneous supramolecular catalysis through immobilization of anionic M4L6 assemblies on cationic polymers," *Journal of the American Chemical Society*, vol. 142, no. 45, pp. 19327–19338, 2020.

[2] X. Gong, F. Wu, R. Xing, J. Du, and C. Liu, "LCBRG: a lane-level road cluster mining algorithm with bidirectional region growing," *Open Geosciences*, vol. 13, no. 1, pp. 835–850, 2021.

[3] M. Fan and A. Sharma, "Design and implementation of construction cost prediction model based on SVM and LSSVM in industries 4.0," *International Journal of Intelligent Computing and Cybernetics, Ahead-Of-Print(Ahead-Of-Print)*, vol. 2, 2021.

[4] J. Wang, X. Li, R. Ruiz, J. Yang, and D. Chu, "Energy utilization task scheduling for mapreduce in heterogeneous clusters," *IEEE Transactions on Services Computing*, vol. 15, no. 2, p. 1, 2020.

[5] J. Jayakumar, B. Nagaraj, S. Chacko, and P. Ajay, "Conceptual implementation of artificial intelligent based E-mobility controller in smart city environment," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 5325116, 2021.

[6] H. Xu, Y. Liu, and W. C. Lau, "Optimal job scheduling with resource packing for heterogeneous servers," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1553–1566, 2021.

[7] L. Li, Y. Diao, and X. Liu, "Ce-Mn mixed oxides supported on glass-fiber for low-temperature selective catalytic reduction of NO with NH3," *Journal Of Rare Earths*, vol. 32, no. 5, pp. 409–415, 2014.

[8] E. P. Goldenberg and C. J. Carter, "Programming as a language for young children to express and explore mathematics in school," *British Journal of Educational Technology*, vol. 52, no. 3, pp. 969–985, 2021.

[9] P. Bachiller, I. Barbecho, L. V. Calderita, P. Bustos, and L. J. Manso, "Learnblock: a robot-agnostic educational programming tool," *IEEE Access*, vol. 8, pp. 30012–30026, 2020.

[10] T. Saito and Y. Watanobe, "Learning path recommendation system for programming education based on neural networks," *International Journal of Distance Education Technologies*, vol. 18, no. 1, pp. 36–64, 2020.

[11] Z. Xia, X. Han, and J. Mao, "Assessment and validation of very-large-eddy simulation turbulence modeling for strongly swirling turbulent flow," *AIAA Journal*, vol. 58, no. 1, pp. 148–163, 2020.

[12] K. Tawackolian and M. Kriegel, "Turbulence model performance for ventilation components pressure losses," *Building Simulation*, vol. 15, no. 3, pp. 389–399, 2022.

[13] R. Huang, "Framework for a smart adult education environment," *World Transactions on Engineering and Technology Education*, vol. 13, no. 4, pp. 637–641, 2015.

[14] Y. Li and F. Xu, "All-phase fast Fourier transform and multiple cross-correlation analysis based on Geiger iteration for acoustic emission sources localization in complex metallic structures," *Structural Health Monitoring*, vol. 21, no. 3, pp. 1235–1250, 2022.

[15] Y. Chen, W. Zhang, L. Dong, K. Cengiz, and A. Sharma, "Study on vibration and noise influence for optimization of garden mower," *Nonlinear Engineering*, vol. 10, no. 1, pp. 428–435, 2021.

[16] Y. Qiang and W. Li, "Accelerated pseudo-spectral method of self-consistent field theory via crystallographic fast Fourier transform," *Macromolecules*, vol. 53, no. 22, pp. 9943–9952, 2020.

[17] A. Bhardwaj, "Machine learning and optimization techniques in major human fatal disorders: a short communication," *International Journal of Computer Applications & Information Technology*, vol. 13, no. 1, pp. 423–427, 2021.

[18] J. Zhang, "Interaction design research based on large data rule mining and blockchain communication technology," *Soft Computing*, vol. 24, no. 21, pp. 16593–16604, 2020.

[19] H. Xiao, S. Ali, Z. Zhang, M. S. Sarfraz, and M. Faisal, "Big data, extracting insights, comprehension, and analytics in cardiology: an overview," *Journal of Healthcare Engineering*, vol. 2021, Article ID 6635463, 2021.

[20] W. Wang, "Deployment and optimization of wireless network node deployment and optimization in smart cities," *Computer Communications*, vol. 155, pp. 117–124, 2020.

[21] I. Awajan, M. Mohamad, and A. Al-Quran, "Sentiment analysis technique and neutrosophic set theory for mining and ranking big data from online reviews," *IEEE Access*, vol. 9, no. 99, pp. 47338–47353, 2021.