2023 International Research Conference of Sri Lanka Technology Campus
Colombo, Sri Lanka. 14th-15th December 2023

Model Parallelism for Efficient GPU Computing in Deep Learning Applications: Comprehensive Review

H.M.S.S. Herath
Computational Intelligence and Robotics Research Lab
Sri Lanka Technological Campus
Padukka, Sri Lanka
sewmih@sltc.ac.lk
https://orcid.org/0009-0008-9702-5576

H.M.K.K.M.B. Herath
Computational Intelligence and Robotics Research Lab
Sri Lanka Technological Campus
Padukka, Sri Lanka
kasunkh@sltc.ac.lk
https://orcid.org/0000-0002-1873-768X

***Abstract*—This comprehensive analysis explores the transformative impact of parallel computing techniques in deep learning. It examines the collaborative endeavors of computer scientists and domain-specific researchers, encompassing a broad spectrum of strategies ranging from conventional data parallelism to cutting-edge methodologies like pipeline and inter-operator parallelism. By democratizing access to high-performance computing resources, these innovations are redefining the landscape of artificial intelligence (AI). The study highlights the considerable enhancements in training efficiency and model accuracy while addressing challenges such as integration complexities and ethical considerations. Additionally, the research investigates the environmental implications of large-scale parallel computing, underscoring the need for sustainable, long-term solutions to minimize its impact. It emphasizes the practical importance of these advancements, particularly in critical sectors such as healthcare and education, where AI-driven innovations hold the potential to revolutionize existing practices. Emphasizing a holistic approach, the analysis advocates incorporating ethical, environmental, and societal considerations in developing AI technologies. Envisioning a future where artificial intelligence is robust but also inclusive and sustainable, this analysis serves as a roadmap for fostering a more accessible, ethical, and environmentally conscious era of artificial intelligence. As the research community continues to push boundaries, this study guides the realization of responsible and impactful AI implementation.**

Keywords—Artificial intelligence (AI), data parallelism, inter-operator parallelism, parallel computing

I. INTRODUCTION

Deep learning methods face several challenges, including computational intensity, high memory requirements, and data parallelism bottlenecks, all of which impede the efficient training of large-scale models. Furthermore, issues such as communication overhead and scalability constraints limit the capabilities of existing deep-learning approaches. By introducing strategies such as parameter shading and task parallelism, model parallelism emerges as a solution to these challenges. The distribution of a model's parameters across different devices optimizes memory utilization and allows for the training of more complex models. In contrast, task parallelism divides computational tasks among devices, reducing the computational intensity of deep learning training. Hybrid approaches combining various parallelization techniques, dynamic computational graphs, and optimized communication all contribute to the effectiveness of model parallelism.

Examining the landscape of different parallel computing models exposes a complex tapestry of strategies and techniques that have transformed the computer industry. Executing several tasks or processes simultaneously is a fundamental idea known as task parallelism. It is frequently used in web servers to handle many concurrent user requests and in scientific simulations to decompose complicated issues into manageable pieces. Contrarily, data parallelism focuses on managing numerous data sets or components concurrently. It is essential in industries like image processing, where processes are evenly performed to individual pixels or frames for increased efficiency [1].

To improve the performance of current CPUs and GPUs, bit-level parallelism takes us deep within computer design fundamentals. Another architectural marvel, instruction-level parallelism, optimizes the execution of machine instructions, enabling contemporary microprocessors to carry out several instructions simultaneously and speeding up program execution [2, 3]. Task farming is widely used in rendering farms for computer-generated imagery (CGI) and distributed computing projects like SETI@home, where volunteers pool their computing capacity and distribute similar jobs among numerous processors or cores. Work is split into numerous phases using the pipeline parallelism method, allowing concurrent execution and data transmission. Digital signal processing and video encoding are applications where real-time data transformation and compression are necessary. Message passing and shared memory parallelism are used to meet the needs of several processes or threads for data exchange and communication [4, 5]. These techniques, which ensure synchronization and data consistency, are crucial in multiprocessor environments, distributed systems, and multithreaded applications.

The foundation of cloud computing, distributed databases, and scientific research initiatives that need significant computational resources is distributed computing, which scales processing over networks and involves several computers or nodes working together to solve complicated problems. Deep learning, scientific simulations, and a wide range of high-performance computing (HPC) jobs are just a few of the tasks that GPU computing finds widespread use in. GPU computing

harnesses Graphics Processing Units' parallel processing power [6, 7].

Choosing a particular parallel computing paradigm depends on several variables, including the task's nature, the available hardware infrastructure, and the required performance levels. The landscape of parallel computing, which enables the effective resolution of complex problems across various areas, is an example of its versatility, significance, and transformational impact on modern computing.

II. Literature Review

The necessity for considerable GPU resources, CPU parallelization, and Neural Network (NN) architectures in contemporary Machine Learning (ML) applications was covered by Goncharo et al. [8]. Due to its lack of multi-GPU training capabilities and adequate parallel CPU data preparation, the Ariadne library—created to solve challenging high-energy physics tracking issues using deep neural networks—faces a particular hurdle.

The authors explain their method for enabling multi-GPU training within the Ariadne library (see Fig. 1) in response to these difficulties. Their approach consists of several crucial elements: effective data caching, parallel CPU-based data preprocessing, and a general ML experiment structure designed for deep neural network model development, training, and inference. The authors also summarize their findings, highlighting the gains in speed and efficiency made possible by the GOVORUN computer resources.

This work improves the Ariadne library's capacity to handle challenging high-energy physics tracking problems by adding multi-GPU training and effective CPU parallelization. This work addresses the crucial need for scaling ML and NN applications.

A cutting-edge technology called PipeDream was created by Harlap et al. [9] to train Deep Neural Networks (DNNs) using GPUs. PipeDream uses a pipeline parallel computing architecture, which spreads computation over numerous computers, in contrast to conventional data-parallel training techniques. This method reduces the high communication-to-computation ratios from working with huge models or networks with limited bandwidth.

With PipeDream, communication overhead is significantly reduced compared to data-parallel training, with up to 95% reductions seen for big DNNs. Fig. 2 shows the high-level workflow of PipeDream. Additionally, it provides continuous processing and communication overlap, assuring maximum GPU utilization. PipeDream carefully allocates DNN layers across the available GPUs to balance workloads and reduce communication needs. Additionally, it uses parameter versioning for backward pass accuracy and round-robin scheduling for forward and backward passes on various inputs to enhance "*time to target accuracy*".

Using PipeDream with different DNNs on different clusters, experiments have shown how successful it is. Compared to conventional data-parallel training approaches, it has been discovered to be up to 5 times quicker in achieving goal accuracy. For large-scale deep learning applications, PipeDream offers improved efficiency, less communication overhead, and faster time-to-accuracy and is a significant development in DNN training.

The DNN training requires a lot of computing and might take days to weeks to finish. Parallel execution using Graphics Processing Units (GPUs) has been a popular strategy to speed up this training. The most common approach, data parallelism, is easier to implement but suffers from high inter-GPU communication costs because of frequent weight synchronization.

Pipelined model parallelism is an alternate strategy that divides the DNN model among GPUs to enable concurrent processing of several mini-batches. Compared to data parallelism, the method proposed by Chen et al. [10] lowers inter-GPU communication costs, but it still struggles with weight staleness. Gradients are calculated using out-of-date weights, which causes training instability and accuracy loss.

The pipelined model, the parallel execution strategy described in [10], maximizes GPU utilization while maintaining training accuracy. This is accomplished using a brand-new weight prediction method called "SpecTrain." Compared to data parallelism on a 4-GPU platform, experimental findings show that this strategy can achieve a tremendous speedup of increase to 8.91 times while maintaining a similar degree of model correctness. In conclusion, the suggested strategy addresses the trade-off between GPU utilization and training accuracy that afflicts current parallelization strategies and significantly improves DNN training efficiency.

Huang et al. [11] introduced GPipe, a novel pipeline parallelism library designed to address the challenge of efficiently scaling DNN capacity for various machine-learning tasks. Scaling up DNN capacity has proven effective in enhancing model quality but often necessitates specialized algorithms or infrastructure when the model size exceeds the memory limits of a single accelerator. These solutions tend to be architecture-specific and lack general applicability across different tasks.

Meanwhile, GPipe provides a practical and task-independent model parallelism solution. This is accomplished by scaling any network represented as a series of layers. Multiple sub-sequences of layers are distributed among various accelerators using GPipe's pipeline parallelism method. This architecture offers the adaptability to grow various networks quickly to much bigger sizes—the sequence of operations given in Fig. 3.

Its main innovation is the batch-splitting pipelining approach GPipe uses, which achieves almost linear speedup when dividing a model over several accelerators. The paper uses two unique projects with various network topologies to demonstrate the benefits of GPipe in practice.

1. Image classification: Using GPipe, an astounding 557 million-parameter AmoebaNet model is trained, and it uses the ImageNet-2012 dataset to achieve a top-one accuracy of 84.4%.
2. GPipe is used to train a single 6-billion-parameter, 128-layer Transformer model on a heterogeneous

corpus covering more than 100 languages for multilingual neural machine translation. This model performs better than all bilingual models, proving the value of GPipe for challenging, extensive multilingual jobs.

In conclusion, GPipe is a vital tool for enhancing the quality and capabilities of machine learning models since it provides a robust solution for effectively scaling deep neural network capacity across various workloads.

The methods for training huge transformer models that Shoeybi et al. [12] presented have shown considerable improvements in Natural Language Processing (NLP) applications. Large models, however, pose difficulties because of memory limitations. The authors of this study presented an effective intra-layer model parallel technique that makes it possible to train transformer models with a massive number of parameters. Notably, their method requires only a few communication operations within native PyTorch and may be easily implemented without requiring modifications to compilers or libraries. Pipeline model parallelism is opposed to and complemented by this method.

The authors demonstrate their methodology and impressive results by leveraging 512 GPUs to train transformer-based models with up to 8.3 billion parameters. Compared to a robust single GPU baseline that can handle 39 TeraFLOPs (about 30% of peak FLOPs), they can handle 15.1 PetaFLOPs of processing power throughout the application with 76% scaling efficiency. The researchers trained an 8.3 billion variable transformers language model (similar to GPT (Generative Pre-Trained Transformer 2)) version 2 and a 3.9 billion parameter model (similar to BERT (Bidirectional Encoder Representations from Transformers)) to show the impact of substantial language models. They emphasize the relevance of optimizing the layer normalization location as the model size grows in BERT-like models.

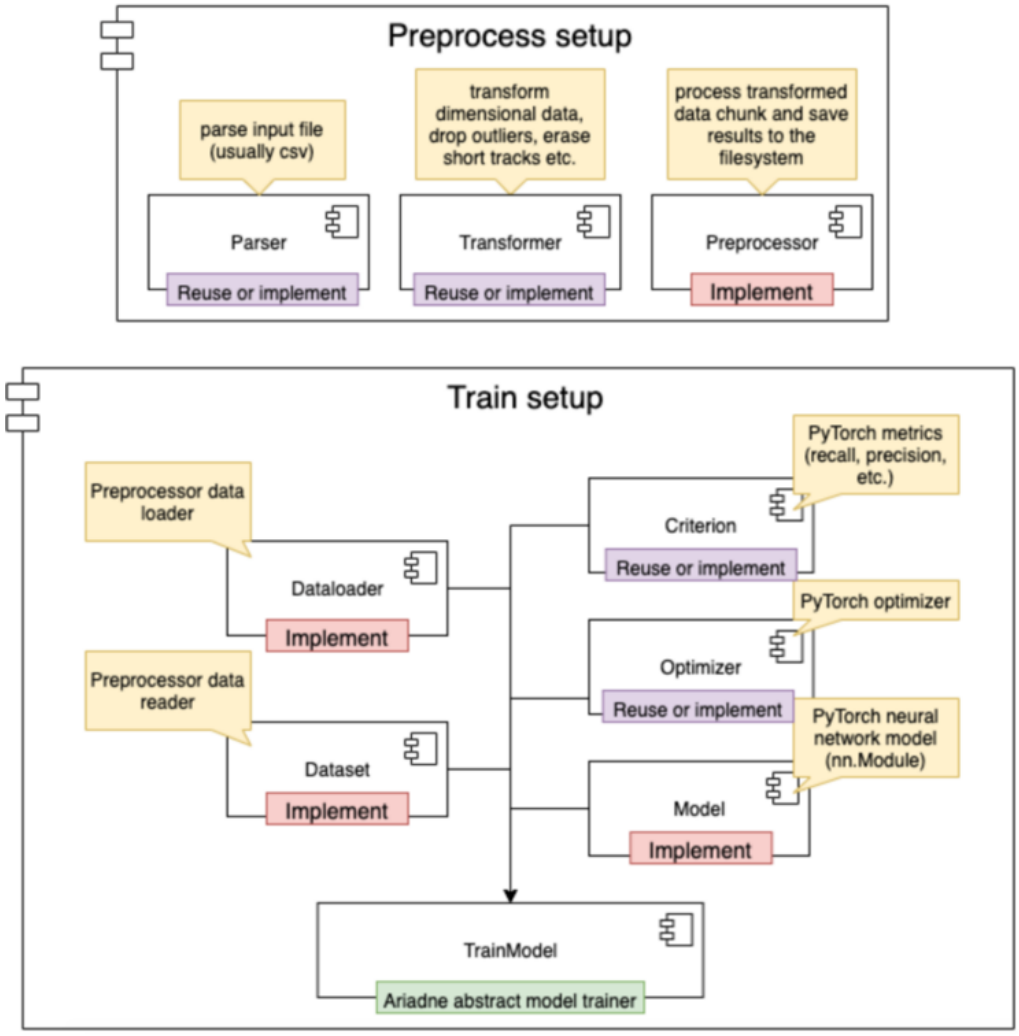


Fig. 1. Process setup and train setup of Ariadne API

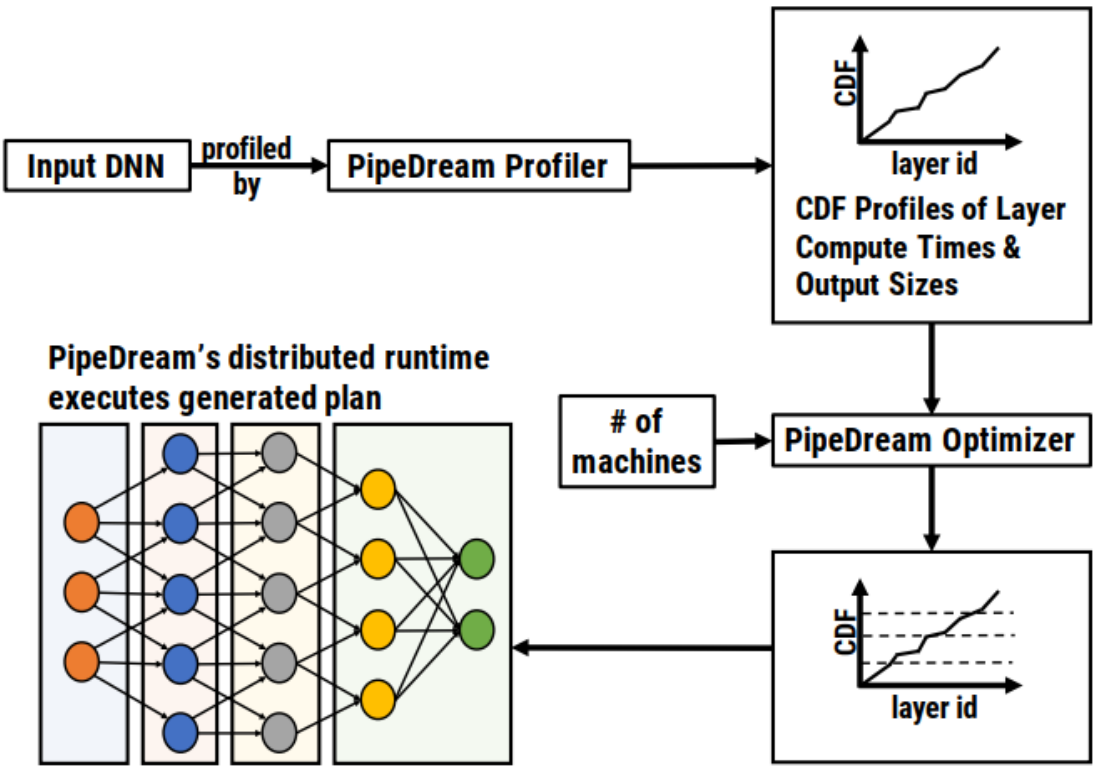


Fig. 2. Pipedream's automated mechanism to partition DNN layers into stages. pipedream first profiles the input DNN to get estimates for each layer's compute time and output size. Using these estimates, pipedream's optimizer partitions layers across available

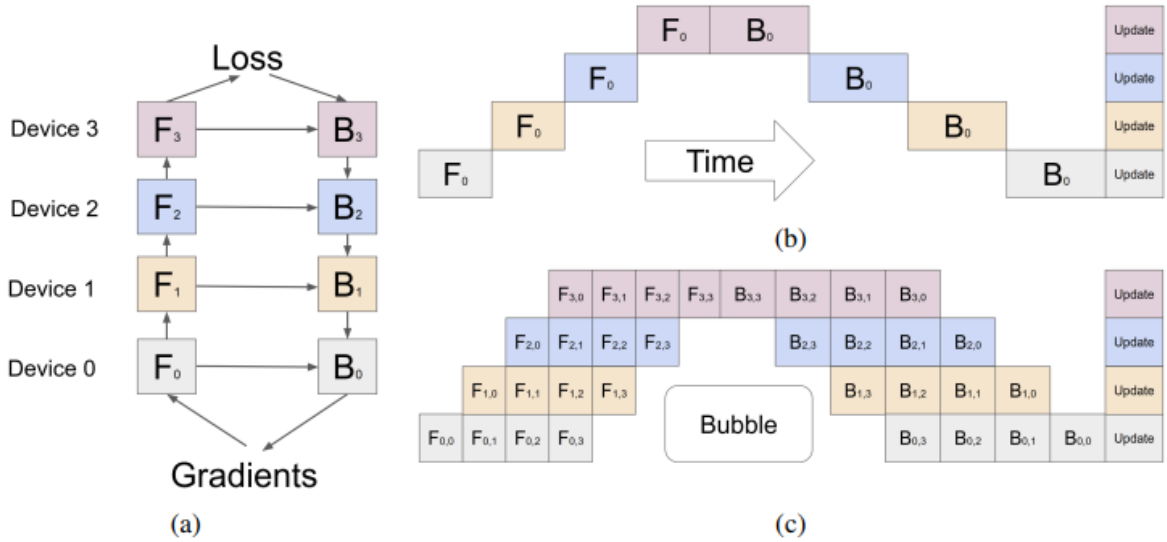


Fig. 3. (a) Illustrates a NN partitioned across four accelerators, (b) The inefficiency of the naive model parallelism strategy due to sequential network dependencies. (c) Introduces pipeline parallelism

On the WikiText103 and LAMBADA datasets, the authors produce cutting-edge results using the GPT-2 model, outperforming the prior best perplexity and accuracy scores. On the RACE dataset, their BERT model achieves advanced accuracy. In conclusion, the research provides methods for effectively training enormous transformer models, exemplifying their efficacy by excellent scaling and attaining cutting-edge outcomes on distinct NLP datasets. These developments may further improve the capabilities of huge language models in natural language processing.

The difficulties of training big DNN models, which have been expanding in size to increase accuracy and quality, were discussed by Park et al. in their study [13]. Such models frequently need to be trained to utilize a heterogeneous cluster of GPUs, including weaker GPUs unsuitable for training alone.

The authors respond to this problem by introducing HetPipe (Heterogeneous Pipeline), a DNN training method that combines data parallelism (DP) with pipelined model parallelism (PMP). In HetPipe, several GPUs create a virtual worker that executes minibatches in a pipelined manner.

Then, data parallelism is used by several virtual workers to improve performance further. The research also offers Wave Synchronous Parallel (WSP), a revolutionary parameter synchronization technique that supports both PMP and DP for virtual workers. Notably, the authors provide convergence evidence for WSP, guaranteeing the effectiveness and efficiency of the training procedure.

Fig. 4 depicts the architecture of the proposed H-node cluster system. Each node has a homogeneous set of GPUs, but the nodes' GPUs (and memory capacity) can be heterogeneous. The efficiency of HetPipe has been demonstrated by experimental findings in a diverse environment. HetPipe allows up to 49% quicker convergence of DNN models than state-of-the-art DP methods. This method significantly improves the effective training of large DNN models on heterogeneous GPU clusters. It enables using a range of GPUs, including less capable ones, to increase the speed and quality of model training.

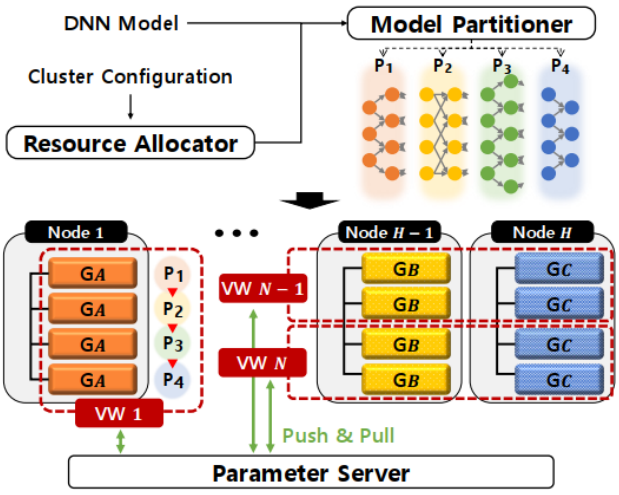
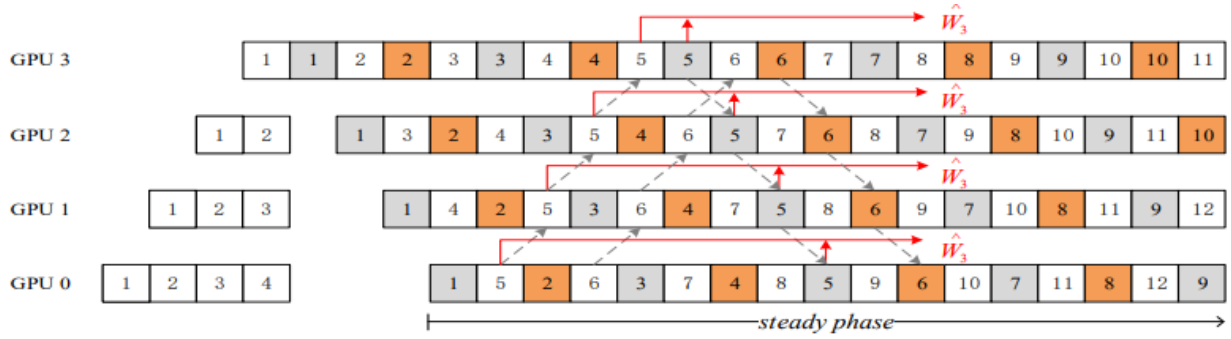
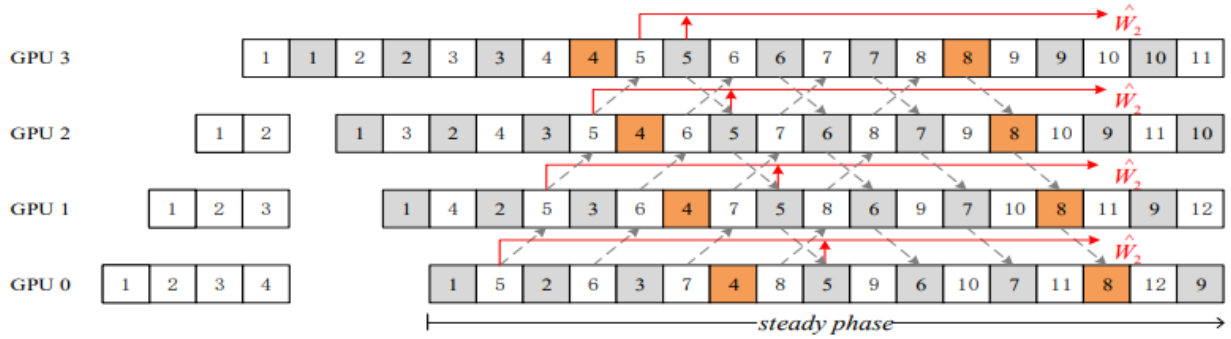


Fig. 4. The architecture of the proposed cluster system



(a) XPipe with $T = 2$



(b) XPipe with $T = 4$

Fig. 5. Illustration of Xpipe workflow on the 4-GPU system. Top: Xpipe workflow with micro batches = 2; Bottom: Xpipe workflow with micro batches = 4, *Adopted from [15]*

In distributed stochastic gradient descent (SGD) training of DNNs, the communication bottleneck problem was first addressed by Ström [14]. Due to the frequent necessity for synchronization of a model replica between compute nodes in data-parallel SGD, which results in a high communication cost, this issue emerges.

The suggested approach addresses this issue by purposefully regulating the rate of weight updates for specific weights inside the model. This differs from the conventional method, which imposes a consistent update rate based on the size of a mini-batch. According to the empirical data in the study, this approach can significantly reduce communication needs during the training of a standard DNN for tasks like acoustic modeling by as much as three orders of magnitude. This technology's decreased communication bandwidth enables scalability to more parallel GPU nodes more effectively than any other known approach. Surprisingly, this increased scalability is attained without compromising the DNN model's accuracy or convergence rate.

This method's noteworthy benefit is enabling training on commodity cloud networking and equipment, making it available and affordable. In conclusion, the study offers a novel approach to the communication bottleneck issue in distributed DNN training, notably lowering communication costs and enabling effective scaling while preserving model convergence and accuracy.

Guan et al. [15] present XPipe, a novel approach for optimizing pipeline model parallelism for training DNNs across multiple GPUs. By introducing an efficient and scalable pipeline model parallelism technique, the study addresses the challenges of scaling DNN training, particularly in the context of large-scale models. The workflow of the XPipe is shown in Fig. 5.

The authors introduce XPipe as a solution to improve the parallelization of DNN training across multiple GPUs in this study. The key innovation is the pipeline parallelism strategy, which divides and processes different segments of the neural network concurrently across various GPUs. XPipe optimizes this process by introducing efficient communication techniques, overlapping computation, and communication. XPipe significantly improves the training efficiency of large-scale DNNs by carefully managing pipeline stages and reducing communication overhead.

The paper provides a thorough technical analysis of the XPipe framework, detailing its architecture and the methodologies used for workload balancing, minimizing communication bottlenecks, and ensuring effective synchronization. The authors present experimental results demonstrating that XPipe outperforms traditional parallelization techniques. Extensive testing has revealed that XPipe achieves remarkable speedups in DNN training while retaining training accuracy, making it a promising solution for large-scale DNN applications. A comparison of HetPipe with other studies is given in Tab. 1.

GEMS [16], which stands for GPU-enabled memory-aware Model-Parallelism System, uses GPUs to address the challenges of large-scale DNN training by introducing memory-aware techniques. The authors' focus in this study is on improving the efficiency of distributed DNN training. They present GEMS as a solution that leverages the computational power of GPUs while keeping memory constraints in mind. The system employs novel model-parallelism techniques, allowing neural network models to be distributed across multiple GPUs. What distinguishes GEMS is its memory-aware approach, in which the system manages memory usage intelligently, ensuring optimal utilization of available resources without compromising performance.

The paper provides a detailed technical overview of GEMS, describing its architecture and the novel memory-aware methodologies used. These techniques allow GEMS to handle large-scale DNNs effectively, making it particularly useful for tasks requiring extensive computational resources and memory, such as deep learning applications in scientific research and artificial intelligence.

Furthermore, the authors present experimental results demonstrating the efficacy of GEMS. GEMS have significantly improved training efficiency and memory utilization in rigorous evaluations, making it a promising solution for accelerating distributed DNN training tasks. The researcher’s findings highlight GEMS’s valuable contribution to high-performance computing, particularly in large-scale machine learning applications.

A groundbreaking analytical model aimed at understanding the complexities of graphics processing unit (GPU) architectures was authored by Hong et al. [17]. The authors acknowledge the increasing importance of GPUs in modern computing and delve into the intricate interplay between memory and thread parallelism within these architectures in this study. They propose an analytical model that captures the nuances of GPU behavior by considering concurrent thread execution and parallel memory access processing. This model considers the dynamic nature of memory access patterns and thread execution, providing a more accurate representation of real-world GPU performance.

The components of their analytical model are meticulously described in the paper, including how it incorporates memory-level and thread-level parallelism awareness. By considering these two types of parallelism, the model provides a comprehensive understanding of GPU execution efficiency, shedding light on the factors that influence performance bottlenecks and throughput limitations.

The authors also validate their analytical model by comparing its predictions to empirical data from fundamental GPU architectures. They demonstrate the model's accuracy in capturing the intricate behaviors of GPUs through rigorous analysis and experimentation, both in terms of memory access patterns and thread execution dynamics.

Zhou et al. [18] acknowledge the growing trend toward billion-scale machine learning models, which present significant challenges due to their massive memory requirements. MPress takes a novel approach to training by focusing on inter-operator parallelism and optimizing memory usage. MPress reduces the memory footprint by carefully orchestrating the data flow between operators in the neural network, making it possible to train large models on multi-GPU servers without excessive memory requirements.

It explains MPress's methodology in detail, emphasizing memory-saving techniques. These techniques include novel inter-operator parallelism strategies allowing more efficient use of available memory resources. MPress enables researchers and practitioners to train billion-scale models on conventional multi-GPU servers by reducing the memory overhead associated with these models democratizing access to large-scale machine learning capabilities.

Furthermore, the authors demonstrate MPress's effectiveness in the paper's experimental results. These experiments show significant memory savings and efficient multi-GPU scaling when training billion-scale models, validating the proposed framework's practical applicability and effectiveness.

Zhang et al. [19]. Introduced the significance of task parallelism in modern computing applications. They are particularly interested in dynamic task parallelism, in which the number and complexity of tasks can vary dynamically during program execution. Traditional parallel processing models face difficulties dynamically managing tasks and allocating computational resources efficiently.

The CPU-assisted GPU thread pool model proposed here introduces a strategic collaboration between CPUs and GPUs. It uses a thread pool mechanism to dynamically assign tasks to threads based on their complexity and computational requirements. The CPU is critical in orchestrating these tasks, distributing them efficiently among available GPU threads. The model ensures optimal utilization of CPU and GPU resources by intelligently managing task allocation and synchronization.

TABLE I. COMPARISON OF HETPIPE WITH GPIPE, PIPEDREAM, AND XPIPE

Features	**Model Parallelism Libraries**			
	GPipe	***PipeDream***	***HetPipe***	***XPipe***
Heterogeneous cluster support	No	No	Yes	Yes
Target large model training	Yes	No	Yes	Yes
Number of workers (virtual)	1	1	No	No
Data parallelism	Extensible	Partition	Virtual Workers	Pipeline Model Parallelism
Proof of convergence	Analytical	Empirical	Analytical	Empirical

The study delves into the technical aspects of this hybrid computational model, revealing information about task scheduling algorithms and coordination mechanisms. The authors present experimental results that demonstrate the efficacy of their method. These experiments show improved performance in managing dynamic task parallelism, which makes the proposed model especially useful for applications with varying computational workloads.

Hong et al. [20] delved into the unique challenges posed by GPUs. It reflects on their previous work on developing an analytical model incorporating memory-level and thread-level parallelism awareness.

The authors revisit their earlier work in this study and provide a retrospective perspective on the analytical model they developed. The model was created to account for the complexities of GPU architectures by considering both memory-level parallelism (MLP) and thread-level parallelism (TLP). Memory-level parallelism refers to the execution of memory operations concurrently, whereas thread-level parallelism refers to executing multiple threads together.

The work discusses the motivations for developing the analytical model, emphasizing the importance of understanding the dynamic interactions between memory access patterns and thread execution behaviors in GPUs. The authors thoroughly explain the model's components and the methodologies used to incorporate memory and thread parallelism awareness. They also discuss how the model has influenced their understanding of GPU performance characteristics, such as memory bottlenecks and thread throughput limitations.

The work [21] emphasizes the difficulties of scaling deep learning models to multiple GPUs, emphasizing the importance of efficient inter-GPU communication and user-friendly training code adaptation. The training library must support such communication, with varying overhead depending on the methods used, which adds to the complications of multi-GPU training. Furthermore, users frequently face the burden of extensively modifying their training code to take advantage of inter-GPU communication. Existing TensorFlow library methods are criticized for their non-negligible communication overhead and the significant code changes required, discouraging many researchers from pursuing multi-GPU training. Horovod, an open-source library designed to address these challenges, is introduced in the paper. Horovod provides efficient inter-GPU communication by reducing ring size, reducing overhead, and requiring only a few lines of user code modification. This makes distributed training in TensorFlow faster and more accessible, addressing the challenges of scaling modern deep learning models.

III. DISCUSSION

In the framework of current machine learning applications, Tab. 2 summarizes the main ideas and contributions from each of the studies mentioned.

Delving into the extensive body of literature on parallel computing for deep learning reveals that the field has undergone a transformative phase marked by innovative strategies to improve the efficiency and scalability of machine learning algorithms. This research has investigated various parallel computing aspects, from traditional data parallelism to ground-breaking techniques such as pipeline parallelism, inter-operator parallelism, and memory-aware model parallelism. These methods effectively address the challenges posed by large-scale DNN models and serve as a steppingstone toward democratizing access to high-performance computing resources.

A key takeaway from this literature is the critical role of collaboration among experts from various domains. The

interdisciplinary nature of these studies, which entails the expertise of computer scientists, engineers, and domain-specific researchers, exemplifies the synergy that occurs when different perspectives collide. This collaborative effort resulted in practical solutions based on theoretical insights, highlighting the importance of a multidisciplinary approach in addressing complex challenges in artificial intelligence.

Furthermore, the literature emphasizes the critical importance of considering the real-world implications of these advances. While the studies show significant improvements in training efficiency and model accuracy, they highlight substantial challenges, particularly regarding integration complexities and ethical considerations. The seamless integration of parallel computing techniques into existing infrastructures necessitates careful planning and concern, particularly in industries that require real-time processing and precision, such as healthcare and finance.

Furthermore, the environmental impact of large-scale parallel computing cannot be ignored. The energy required to train models grows significantly as they become more complex. This reality necessitates a critical examination of eco-friendly practices and the development of energy-efficient computing solutions. Collaboration between researchers and industry experts is essential to developing green computing strategies, ensuring that adverse environmental impacts do not undermine the benefits of parallel computing.

Literature not only provides valuable insights into the democratization of artificial intelligence, but it also contextualizes it within practical applications. These studies have made it possible to train sophisticated models on standard hardware by optimizing parallel computing techniques, making advanced machine learning capabilities accessible to a broader audience. This democratization has far-reaching implications, particularly in fields such as healthcare, where AI-driven diagnostics and personalized treatments can transform patient care, and education, where intelligent tutoring systems can significantly improve learning experiences. This multifaceted impact highlights parallel computing's transformative power in shaping a more accessible and impactful future for artificial intelligence applications across multiple domains.

Model parallelism has demonstrated its effectiveness in improving training efficiency and model accuracy across a wide range of domains. Notably, in natural language processing, BERT has used model parallelism to distribute its massive parameters efficiently, achieving state-of-the-art results in tasks such as question answering and sentiment analysis. Similarly, Open AI's GPT-3 uses both data parallelism and model parallelism, distributing model segments across GPUs to achieve efficient training and the generation of contextually relevant text on various prompts. Model parallelism optimizes the training of deep Convolutional Neural Networks (CNNs) in computer vision, particularly for large-scale image classification tasks, demonstrating improved efficiency and scalability.

Model parallelism is implemented in distributed deep learning frameworks such as Horovod, reducing communication overhead and accelerating the training of large-scale models in tasks such as image and speech recognition. Furthermore, model parallelism effectively trains models representing complex policies in reinforcement learning, particularly in applications such as robotics. These examples demonstrate model parallelism's versatility and success in addressing challenges associated with large-scale deep-learning models, improving training efficiency and model accuracy.

Model parallelism integration in deep learning frameworks poses complex challenges requiring nuanced solutions. One major challenge is partitioning complex computational graphs across multiple devices, necessitating heuristic-based approaches and optimization algorithms for optimal distribution. Inter-device communication introduces latency and overhead, necessitating efficiency-enhancing strategies such as gradient compression and specialized communication libraries such as NCCL [22]. The issue of balancing synchronization in training, which is critical for maintaining model integrity, is addressed using hybrid approaches that combine synchronous and asynchronous methods. Adaptive techniques like model slicing and dynamic graph construction are required because of the dynamic nature of modern network architectures, such as recurrent neural networks and attention mechanisms. Specialized tools like TensorFlow’s distributed tracing and Horovod's built-in profiling capabilities debug and profile distributed model parallel systems. It is critical to ensure user-friendly APIs for widespread adoption, and frameworks such as PyTorch and TensorFlow are evolving to provide high-level abstractions, making model parallelism more accessible. In practice, the convergence of algorithmic advances, specialized libraries, and user-friendly tools contributes to the efficient integration of model parallelism, constantly refining the landscape of distributed training in deep learning frameworks.

TABLE II. KEY FINDINGS OF THE STUDY

Study	Key Findings
Goncharo et al. [8]	They addressed the need for multi-GPU training capabilities and effective CPU parallelization within the Ariadne library for deep learning.
Harlap et al. [9]	They introduced PipeDream, a pipeline parallel computing architecture, to reduce communication overhead and maximize GPU utilization.
Chen et al. [10]	They proposed a pipelined model parallelism strategy, including the "SpecTrain" weight prediction method, for enhanced DNN training efficiency.
Huang et al. [11]	They Developed GPipe, a pipeline parallelism library for effectively scaling DNN capacity across various machine-learning tasks.
Shoeybi et al. [12]	They Presented techniques for training large transformer models, showcasing substantial improvements in Natural Language Processing applications.
Park et al. [13]	They Introduced HetPipe, a novel DNN training method that combines data parallelism with pipelined model parallelism for heterogeneous GPU clusters.

Study	Key Findings
Ström [14]	Through an innovative weight update strategy, she addressed communication bottleneck issues in distributed stochastic gradient descent (SGD) training.
Guan et al. [15]	They Developed XPipe, an efficient and scalable pipeline model parallelism technique for training DNNs across multiple GPUs.
A. Jain et al. [16]	GEMS, a GPU-enabled memory-aware Model-Parallelism System, enhances large-scale distributed deep neural network training by managing memory usage, incorporating novel techniques, and leveraging GPU computational power.
Hong et al. [17]	They proposed an analytical model for understanding GPU architectures, considering memory- and thread-level parallelism dynamics.
Zhou et al. [18]	They Introduced MPress, a memory-efficient training approach emphasizing inter-operator parallelism for large-scale DNN models.
Zhang et al. [19]	They highlighted the significance of task parallelism, particularly dynamic task parallelism, and the collaboration between CPUs and GPUs.
Hong et al. [20]	Respond to the challenges posed by GPUs by revisiting their analytical model, emphasizing the importance of memory-level and thread-level parallelism awareness, and demonstrating its accurate prediction of GPU-based system behavior through validation against real-world implementations.
Sergeev et al. [21]	The article discusses the challenges of training deep learning models on multiple GPUs, introducing Horovod, an open-source library that simplifies distributed training in TensorFlow, enhancing model performance.

IV. Conclusion

The literature review emphasizes parallel computing techniques' transformative impact on deep learning and computational science. The transition from traditional data parallelism to novel methods such as pipeline parallelism, inter-operator parallelism, and memory-aware model parallelism represents a significant step forward in large-scale machine learning tasks. These methods improve DNN training efficiency and democratize access to complex models, allowing for exploration in areas such as natural language processing and scientific simulations. However, challenges remain, necessitating ongoing research to integrate these techniques seamlessly into existing infrastructures and address emerging ethical and environmental concerns related to large-scale computing. Collaboration, ethics, and sustainability are critical in realizing parallel computing's potential for societal improvement and human knowledge advancement.

Reviewing the extensive literature on model parallelism in GPU computing reveals significant research gaps. It is critical to optimize techniques for heterogeneous GPU architectures to use GPUs with varying capabilities efficiently. Scalability studies are critical for investigating the limits of model parallelism as deep learning models grow, as well as trade-offs between model size, GPU numbers, and communication overhead. Due to the lack of dynamic workload management strategies, adaptive model parallelism techniques that can distribute model layers based on varying computational requirements are required.

Model energy-efficient parallelism is essential, prompting research into techniques that optimize energy usage while maintaining accuracy. Protocols enabling seamless integration of model parallelism into popular deep learning frameworks are required to address standardization and compatibility issues across frameworks. There is a lack of empirical studies across a wide range of real-world applications, emphasizing the need for practical implementations to assess model parallelism's performance, challenges, and domain-specific enhancements. It is critical to close these gaps if model parallelism in GPU computing is to progress and become widely used.

Model parallelism is revolutionizing various aspects of medical applications in the healthcare sector. Model parallelism is used in medical imaging to improve the accuracy and efficiency of deep learning models, resulting in more precise diagnoses and treatment plans. The technology speeds up drug discovery processes by more accurately predicting molecular interactions, potentially leading to the development of new medications. Model parallelism is also used in disease prediction based on comprehensive patient data, genomics, and clinical records, promoting early interventions and personalized treatment strategies. Model parallelism causes transformative changes in education, particularly in personalized learning. Model parallelism-based adaptive learning models cater to individual student needs and learning styles, increasing engagement and academic performance. Natural language processing applications in education, such as automated grading and language tutoring, are powered by technology, providing students with timely and personalized feedback. Furthermore, model parallelism in educational research facilitates the training of sophisticated models to understand learning behaviors and predict student outcomes, enabling evidence-based strategies for improved learning experiences. These applications highlight the tangible benefits of model parallelism in healthcare and education, providing unprecedented accuracy, efficiency, and personalization advances.

References

[1] R. Tudoran, A. Costan, and G. Antoniu, “OverFlow: Multi-site aware big data management for scientific workflows on clouds,” IEEE Transactions on Cloud Computing, vol. 4, Art. no. 1, 2016

[2] H. Sharma et al., “Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network,” 2018, pp. 764–775.

[3] S. Ghodrati, H. Sharma, C. Young, N. S. Kim, and H. Esmaeilzadeh, “Bit-parallel vector composability for neural acceleration,” 2020, pp. 1–6.

[4] Z. Li et al., “TeraPipe: Token-level pipeline parallelism for training large-scale language models,” M. Meila and T. Zhang, Eds., PMLR, 2021, pp. 6543–6552

[5] C.-H. Chiu and T.-W. Huang, “Composing pipeline parallelism using control taskflow graph,” Minneapolis, MN, USA: Association for Computing Machinery, 2022, pp. 283–284.

[6] M. Hojnacki et al., “Parallel computation of gröbner bases on a graphics processing unit,” Arabnia, Hamid R, L. Deligiannidis, M. R. Grimaila, D. D. Hodson, K. Joe, M. Sekijima, and F. G. Tinetti, Eds., Springer International Publishing, 2021, pp. 417–432.

[7] A. Ruokamo, “Parallel computing and parallel programming models: application in digital image processing in mobile systems and personal mobile devices,” 2018.

[8] M. Manual, “Heavy Metals, Nitrogen and POPs in European Mosses: 2020 Survey.”

[9] A. Harlap et al., “Pipedream: Fast and efficient pipeline parallel dnn training,” arXiv preprint arXiv:1806.03377, 2018.

[10] C. Chen, C. Yang, and H. Cheng, “Efficient and robust parallel dnn training through model parallelism on multigpu platform,” arXiv preprint arXiv:1809.02839, 2018.

[11] Y. Huang et al., “GPipe: Easy scaling with microbatch pipeline parallelism,” arXiv preprint arXiv:1811.06965, 2018.

[12] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, “Megatronlm: Training multibillion parameter language models using model parallelism,” arXiv preprint arXiv:1909.08053, 2019.

[13] J. H. Park et al., “{HetPipe}: Enabling large {DNN} training on (whimpy) heterogeneous {GPU} clusters through integration of pipelined model parallelism and data parallelism,” in 2020 USENIX Annual Technical Conference (USENIX ATC 20), 2020, pp. 307–321.

[14] N. Ström, “Scalable distributed DNN training using commodity GPU cloud computing,” 2015.

[15] L. Guan, W. Yin, D. Li, and X. Lu, “XPipe: Efficient pipeline model parallelism for multiGPU DNN training,” arXiv preprint arXiv:1911.04610, 2019.

[16] A. Jain et al., “Gems: Gpuenabled memoryaware modelparallelism system for distributed dnn training,” in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2020, pp. 1–15.

[17] S. Hong and H. Kim, “An analytical model for a GPU architecture with memorylevel and threadlevel parallelism awareness,” in Proceedings of the 36th annual international symposium on Computer architecture, 2009, pp. 152–163.

[18] Q. Zhou *et al.*, “MPress: Democratizing Billion-Scale Model Training on Multi-GPU Servers via Memory-Saving Inter-Operator Parallelism,” Feb. 2023, doi: https://doi.org/10.1109/hpca56546.2023.10071077.

[19] S. Zhang, T. Li, Q. Dong, X. Liu, and Y. Yang, “CPU-assisted GPU thread pool model for dynamic task parallelism,” Aug. 2015

[20] S. Hong and H. Kim, “Memorylevel and threadlevel parallelism aware gpu architecture performance analytical model,” 2009.

[21] A. Sergeev and D. Balso, “Horovod: fast and easy distributed deep learning in TensorFlow,” arXiv.org, 2018. https://arxiv.org/abs/1802.05799 (accessed Nov. 25, 2023).

[22] “NVIDIA Collective Communications Library (NCCL),” NVIDIA Developer, 2023. https://developer.nvidia.com/nccl (accessed Nov. 25, 2023).