Parallel Computing with GPU: An accelerator for Data-Centric High Performance Computing

Mohammed Siddique
Faculty of Information Technology
Majan University College
Muscat, Oman
muhammed.siddique@majancollege.edu.om

Mohammed Waleed Ashour
Faculty of Information Technology
Majan University College
Muscat, Oman
mohammed.ashour@majancollege.edu.om

2024 1st International Conference on Innovative Engineering Sciences and Technological Research (ICIESTR) | 979-8-3503-4863-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICIESTR60916.2024.10798202

***Abstract*— In recent years, the demand for high-performance computing in various disciplines of ICTs has affected the dependency on single core processes due to its decline in efficiency. Whereas, this requirement of advance in processing power has seen the emergence of Graphical Processing Units (GPU). In this paper, the role of GPUs in parallel computing along with its advantages, limitations and performance in various domains have been presented. The massive parallelism feature of GPUs can break down complex problems into smaller bits of computation with greater efficiency. The ability to compute complex distributed processes is ideal for implementing Machine Learning (ML) and Deep Learning (DL) algorithms. The evaluation of GPUs performance in medical physics, large scale data mining, CUDA workload analysis and applications of ML and DL has been deliberated on the high-performance computing. As GPUs continue to evolve, processing performance and efficiency should improve, and stand out as a cost-effective and powerful solution for optimizing faster and reliable large-scale data processing. Future research is expected to optimize GPU architectures for developing computing paradigms to keep them relevant in high-performance computing's fast-changing industry.**

***Keywords* – Parallel Computing, GPU, High Performance Computing, Machine Learning, Deep Learning.**

I. Introduction

Graphical Processing Unit (GPU) enhances the performance of higher workloads by accelerating the process. In parallel computing, this is achieved by leveraging the computing capacity of the GPU using concurrent computations. These workloads include processing large image datasets, scientific simulations, data mining, and implementation of Deep Learning (DL) and Machine Learning (ML) algorithms. This in contrast with the Central Processing Unit (CPU) which is not compatible with handling higher workloads. Therefore, GPU is technically designed as an ideal option to handle such workloads [1]. Further, the use of GPUs can significantly accelerate application performance contributing towards an efficient and robust solution [2]. This paper emphasizes on GPUs parallel processing capability, effectiveness, performance optimization. When the applications require high computation power over a longer period, the investment on GPUs will be more cost-effective rather than registering cloud services. GPUs and its relevance in based on computing disciplines and recommendations on various techniques are provided.

The optimization techniques based on Artificial Intelligence (AI) support in finding solutions to undetermined issues. The key benefit is to achieve accuracy with minimal errors during calculations [3]. These optimizers will be required during complex computations such as Heuristic Optimization technique and forecasting energy consumption based on long short-term memory network [4]. The sporadic task model shown in Fig. 1 is commonly deployed in parallel computing is categorised with as an execution requirement being the first parameter, a period as the second parameter and the relative deadline as a third parameter can be utilized in the elucidation of sequence of tasks.

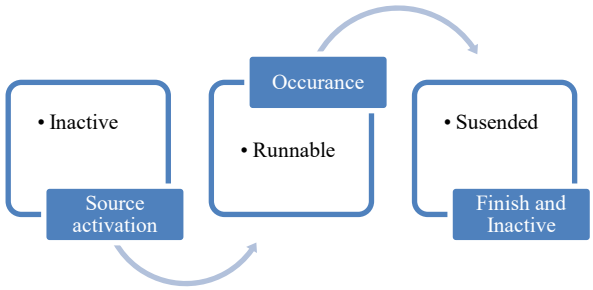

Fig 1. Sporadic Task Model [5].

This model is capable of real-time processing with an enhanced performance that recurs at randomly. The tasks are denoted by

$$T_i = (e_i, g_i, d_i) \quad (1)$$

Where g_i represented minimum separation between two successive occurrences, the d_i represents relative deadline and the e_i represents the worst case execution.

The hyperspectral imaging (HIS) technique utilizes thousands of small spectral bands which need to be collected and further analyze electromagnetic spectrum of their objects or scenes. This is an exhaustive process and requires high-performance computing operation. GPUs can accelerate the implementation of this technique facilitating classification of a large number of hyperspectral images and analysing them by using ML algorithms. This process examines the attributes of each image pixel and categorizes them.

This paper is categorized with Section I covering the introduction and Section II providing a detailed literature review on ML algorithms, DL algorithms, medical physics, Large Scale data mining and Simulations for CUDA workload performance analysis with GPUs. The Section III describes methodologies that include Global Scheduling Heuristics and OpenMP applications on multi-core architectures. Section IV covers discussion on various approaches to maximize performance, boost performance, and parallelize in multicore architectures. Finally, Section V has the conclusion and future recommendation.

II. Related Work

The integration of Graphics Processing Units (GPUs) as accelerators for data-centric applications has been becoming increasingly common in recent advancements in High-

979-8-3503-4863-7/24/$31.00 ©2024 IEEE

Performance Computing (HPC). This section discusses the latest developments, trends, and perceptions in this domain.

A. Machine Learning Algorithms and GPUs

During the past few years, GPUs and ML techniques have revolutionized as an integrated model in the field of computational intelligence. As ML applications need more computational power, GPUs become essential. Noticeably, ML techniques require parallel computing to accomplish complex computational tasks in mathematics. This integration has accelerated machine learning model training and implementation.

GPUs are crucial to modern computing, according to Dally et al. (2021). GPUs run the finest supercomputers and are crucial to Deep Learning. They form the computational backbone of self-driving automobiles and robots [6]. Since NVIDIA's GeForce 256, specialized graphics processors (GPUs) have become programmable entities that can handle floating-point calculations and real-time graphics operations. This makes them appealing for scientific computations. Scientists programmed vertex and fragment shaders on early programmable GPUs to do sophisticated calculations. Advanced programmability, double-precision floating-point arithmetic, and robustness were added to GPUs to meet scientific research needs. This confirmed their importance in scientific computing.

Madiajagan et al. (2019) also showed that GPUs are essential for deep learning computing [7]. DL workloads like complicated matrix multiplications and operations run faster on GPUs due to their massive parallelization. GPUs can have thousands of cores for parallel computation. GPUs have tremendous computational capabilities, but they struggle with long training times and limited memory, especially as neural networks grow. These issues are addressed by studying distributed parallel computing solutions to improve training times and memory utilization. They also concentrate on hardware engine architectures to reduce data size. The latest Tesla GPU has 16 GB of memory, but the memory capacity of commonly used GPUs are important. Thus, neural network design must be customized to fit within GPU memory, highlighting a potential constraint that requires deliberate research to enhance computational intelligence.

Tan et al. (2021) invented ARENA, an Asynchronous Reconfigurable Accelerator Ring Architecture, which transformed HPC and data centers [8]. ARENA's architecture and programming allow asynchronous tasking across reconfigurable nodes and use coarse-grained reconfigurable arrays, making it unique. This technology allowed data-specific processing and dynamic accelerator building during runtime. Task tokens were disseminated via a high-speed ring network in ARENA's distributed data storage system.

In the other hand, GPU parallel computing can efficiently help to improve computational efficiency and processing speed for complex steel surface inspection tasks, as shown in a study conducted by Ashour et al. (2023) [9]. This technique tends to fasten the image analysis and processing phase such as the feature extraction process. GPUs can parallelize multiple tasks, which assists in handling the complex algorithms and computations needed to thoroughly inspect steel surfaces. Machine vision inspection systems can be more efficient and effective due to faster image analysis. The GPU's capacity to efficiently analyze large datasets suits high-performance computing's data-focused nature. It is powerful enough to manage the large amounts of data required in steel surface studies. Fig. 2 shows that Parallel GPU-based techniques may handle noisy images quickly and effectively, even for large datasets [10].

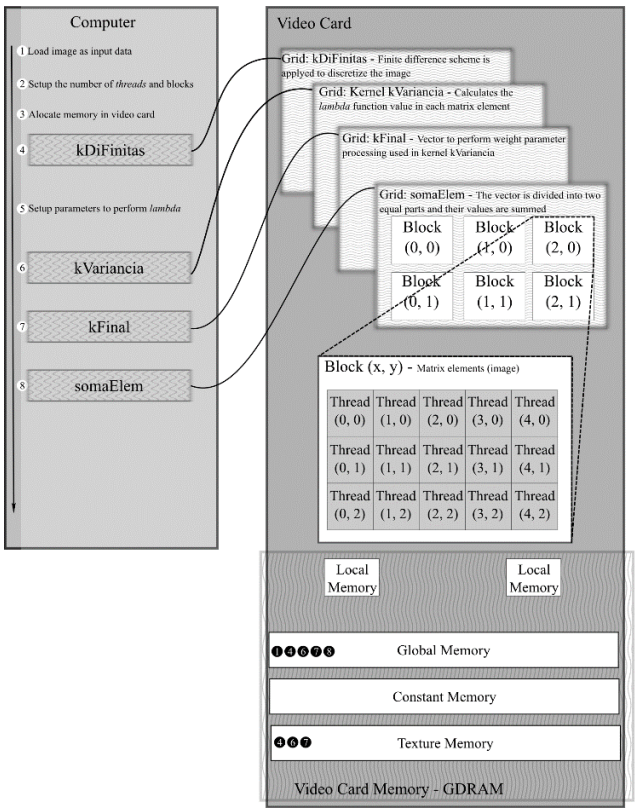

Fig 2. Parallel GPU-based Implementation for image smoothing based on a variation model using the CUDA architecture [11].

DL's ascent was spurred by accelerator technologies, with the GPU becoming the favored solution for DL applications because to its unique properties. Mittal and Vaishay (2019) comprehensively analyzed architectural and system-level solutions to improve GPU-based deep learning applications [11]. The survey comprised inference and training scenarios using a single GPU and many GPUs in distributed systems. By describing similarities and contrasts, the study revealed key characteristics of diverse approaches. This survey helped novices and experts in machine learning, processor design, and high-performance computing understand the evolving landscape.

Maksum et al. (2022) illuminated topology optimization methodologies and the necessity of GPU and ML-based acceleration [12]. Parallel computing employing GPUs, which accelerate data-centric HPC activities, became the emphasis as we explored high-performance computing (HPC). Maksum et al. showed that GPUs in parallel computing with cutting-edge computational optimization formed a dynamic synergy. As mentioned above, this paradigm shifts increased application performance and broadened data-centric high-performance computing. GPU-based parallel computing proved powerful during this expedition, enabling computational intelligence and data-driven scientific research advances.

B. Medical physics and GPUs

In medical physics, where large datasets and complex simulations are common, these can be parallelized for significant speed-ups and allows the processing of large amounts of data in a relatively short amount of time [13]. Additionally, GPUs can often perform these tasks with less power consumption and a smaller physical footprint, making it a more resource-efficient choice. There are several APIs

(Application Programming Interfaces) available for programming GPUs such as CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language). The high performance computing applications can be created using the proprietary CUDA APIs that possess parallel processing capability. In contrast, OpenCL is a preferable open standard cross platform which can be used to program GPUs. This is compatible with most of the hardware vendors and flexible in programming parallel processors such as CPUs and GPUs. The individual Video Random Access Memory (VRAM) stores the required data to render graphics and complex parallel computations. Using these VRAMs is vital to speed-up the large dataset processing.

C. GPUs in Large Scale Data Mining

The trends in data and identification of patterns based on feature extraction and classification can be analyzed using GPUs. In data mining, these trends pave the way for preparing informed marketing strategies and can be achieved utilising GPU frameworks. An optimized implementation of DL techniques such as Convolutional Neural Networks (CNNs) can be achieved using the CUDA Deep Neural Network library (cuDNN) framework from NVIDIA. The ML techniques can implement data mining algorithms and the GPU acceleration is supported by open-source TensorFlow and PyTorch ML frameworks. These frameworks can perform large scale data mining tasks on various hardware platforms [14]. GPUs can perform parallel operations to accelerate image annotation and recognition tasks in a shorter amount of time. However, they require larger datasets of images in order to analyze and classify into different categories.

D. GPU Simulations for CUDA workload performance analysis

The performance analysis and optimization of GPU code on CUDA workloads can be implemented along with testing and debugging using simulation tools such as General-Purpose GPU (GPGPU-Sim) and NVIDIA Visual Profiler [15]. These simulation tools provide total time consumed and resource details for each set of GPU program code of CUDA application. The shared memory details and the instruction pipeline are fetched from the process. In addition to the performance analysis and optimization techniques, these tools are useful in identifying the characteristics of individual kernels and their functions. For example, the debugging and profiling of CUDA and OpenCL applications can be performed using AMD CodeXL suite. Similarly, when running the CUDA scientific or engineering program workloads, the performance measuring features from CUDA Software Development Kit (SDK) such as Rodinia can be used for benchmarking the performance of GPU. It includes a variety of benchmarks for tasks such as image processing, data mining, and molecular dynamics. The Black-Scholes SDK measures the performance of a GPU when running a financial modeling algorithm and N-Body SDK measures the performance of a GPU when running a physics simulation of a system of interacting particles. They test the performance of a GPU when running tasks that involve large amounts of data parallelism. Similarly, a Matrix Multiplication SDK is a simple benchmark that can test the performance of a GPU when running tasks that involve heavy arithmetic calculations. In conclusion, the review emphasized the idea that GPU acceleration improves data-centric HPC. Research shows that GPUs improve computer performance, from parallel designs to their use in many fields.

III. Methodologies

In HPC, the image processing, complex and scientific calculations require concurrent computations. This is achievable using GPUs as accelerators. Song. D et al. (2023) proposed a flexible image processing LiGo framework which accelerates streamlined audio/video processing framework using GPU in industrial applications [16]. By using Bi-directional recurrent neural networks (BiRNN) based on GPU virtualization, the processing of high resolution videos can be efficiently compressed [17]. As illustrated in Fig. 3, the general purpose parallel computing in GPUs using OpenCL and CUDA APIs enables the coding with python programming and the TensorFlow framework offloads computation.

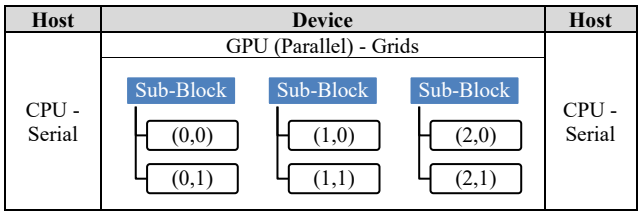

Fig 3. GPU (CUDA) Architecture – Parallel Computing.

In addition to the supported GPU architecture, the processed data and the problem specification are one of the main factors affecting the GPU performance. The OpenMP allows a GPU compilation scheme directly that leverages the target offloading interface [18]. This scheme requires a program wrapper layer, commands, and the C/C++ libraries for GPU implementation.

The GPUs provide results with higher accuracies when ML or DL algorithms such as Random Forest (RF) and Convolutional Neural Networks (CNN) are implemented on complex computations such as image classification and natural language processing [19]. However, the hardware implementation is quite expensive. Field Programmable Gate Arrays (FPGAs) is a promising type of hardware with lower energy consumption towards implementing SNN, an alternative solution [20]. FGPAs support flexible parallel processing and are preferred among low cost digital hardware systems with lower time consumption.

A. Global Scheduling Heuristics

The scheduling for HPC systems is prepared depending on the heterogeneity, computation supported devices that include GPU accelerators and algorithm approaches. This is defined as multiple stages addressing management and monitoring of processes on devices on a network of devices such as clusters, virtualized clouds or computational grids. At times, the tasks can be consolidated and perform parallel as well to achieve efficiency and saving energy [21]. This approach requires accuracy in terms of measuring executive time and energy consumption. The computation on CPU cores or GPU accelerators support measuring of execution time [22]. Chen et al. (2016), proposed a Quantum Inspired Algorithm (QHA) which can speed up the heuristic search process. This empirical approach supports managing of suitable heuristics for energy aware scheduling providing enhanced results for various applications [23].

Kiran et al. (2015), proposed four global scheduling algorithms to address the challenges of parallelizing and scheduling programs in multi-core environments. They aim to achieve power based higher performance, scalability and lower communication cost [24]. The Height Instruction Count Based (HIB) scheduler is a linear time algorithm to schedule

SDG's sub-block in the core by following an effective strategy prioritizing the height and the instruction count. The Dependent Sub-Block Based (DSB) algorithm identifies highest schedule latency for every possible path whereas, the Maximum Dependency Sub-block (MDS) follows minimum execution time and maximum dependencies within the sub-block's nodes to allocate it on different cores. The Longest Latency Sub-Block First (LLSF) algorithm considers latency alone while scheduling the sub-block.

The performance of multi-core systems is calculated in terms of speedup using Amdahl's law [25]. The power consumption and energy efficiency of multiple cores with the fraction of power k consumed by an individual core in idle state is calculated using the Woo-lee model [26]. During parallel computing, a core processor represented by n is considered to consume n units of power. The average power usage in terms of watts is calculated using the formula below.

$$W = \frac{1+(n-1)k(1-f)}{(1-f)+f/n} \qquad (2)$$

The computation fraction that could be parallelized is denoted by f, the execution of sequential programs is represented by (1-f) and the execution of parallel programs is calculated by f/n.

B. OpenMP and Performance Analysis

The limitation of applications run-time is a challenge when tuning the accelerated sections of GPU applications for optimizing performance. The Record-Replay (RR) approach facilitates optimization of OpenMP GPU applications by dedicating individual GPU kernels as executable sections [27]. The smaller program codes can be processed and auto-tuned without any resource requirements or dependencies thus making massive parallelisation. To identify the parameters for an ideal kernel process, a scalable Bayesian optimization technique is implemented increasing speedup of processes.

OpenMP is one of the parallel programming models which can improvise performance. OpenMP's advanced directives features and functionalities support specific device computation. The load balancing at the thread and process levels is key to the parallelization of OpenMP applications. With the dynamic core binding (DCB) method, the consumption of energy and the computation time can be optimized by allocating an unspecified number of cores to each process and addressed at the thread level [28]. OpenMP applications performance is enhanced with this approach along with reduction of core usage without an increase in time computation.

IV. Discussion

Multi-core processors can exploit the parallelism to maximise the performance of applications in process. Based on the frequency of instructions per cycle, the performance is calculated. Every core from this architecture follows the Multiple Instruction Multiple Data stream and executes each of their thread processes on different processors but using shared memory [29]. In cluster computing, a network of computers, memory storage and other devices are connected that communicate with each other and share resources to perform a common task. Each of these devices are referred to as nodes and consist of one or more CPU or GPU. This type of network can perform large-scale computations and hence, they can considerably speed up the execution of processes. The data processing and computation workloads are distributed among the several nodes using grid computing. The GPUs parallel processing capability is effective and can extensively speed up processing. Hence, the utilization of GPU libraries is comparatively easier to configure and one of the most preferred choices in parallel computing.

The hardware accelerators benefit when reconfigurable logic is implemented with FGPAs. This configuration is based on integrated circuits supporting acceleration of parallel blocks under execution. Both CPUs and GPUs consume higher power consumption when utilising cache hierarchies by computing on cores executed on a higher frequency. This is contrary to the fact that FGPAs running on heterogeneous platforms provide efficient computation [30]. In certain cases, where the embedded platforms deploy vision kernels, the accuracy and the efficiency of GPUs are outperformed by FGPA [31]. A simple experiment to compare frame reduction on the vision kernels depicted that the GPU achieved frame reduction ratio of 1.1 – 3.2x whereas it was outperformed by the FGPA with 1.2 – 22.3x frame reduction ratio. The comparison was conducted between the Xilinx Virtex 7 FPGA and a 28nm Nvidia K40c GPU. When comparing GPU and FPGA, the relationship between the run-time and micro-architectural factors is analyzed [32]. This is identified using the formula 2 given below.

$$run - time = \frac{\text{tOps}*\text{Overhead\_factor}}{\text{pipe\_OPC}*e\_para\_factor*f} \qquad (3)$$

Where f is the frequency of the GPU and FGPA kernels clock and tOps is the number of operations at the algorithm level. These algorithm steps include multiplication and sum operations and are executed by the kernel. The array indexing is not considered as the total number of operations are same for the implementation for GPU as well as for FGPA kernel.

Edge GPU in Edge computing is used to enhance the processing efficiency moving the AI computing power towards data at the Edge of the network. The processing and other service operations are streamlined by leveraging the Internet of Things (IoT) sensors-based generated data and analysing it.

Multi-core OpenMP real-time scheduling model works on concurrent events with distributed segments consisting of several synchronized parallelized threads [33]. This concept supports a three parameter sporadic task model used in parallel programming languages and preferred for its real-time processing capability with higher performance. OpenMP supports thread-level locking and high-level compiler directives which relates to low-level and high-level synchronization respectively. These synchronization techniques secure data and implements request restrictions.

V. Conclusion

The acceleration and efficiency of data-centric HPC using GPU is among the most preferred approaches in parallel computing. The implementation of DL models on computer vision, massive neural networks and natural language processing applications requires training on vast volumes of data. This can be achieved by using GPUs as they are well-suited for large scale applications requiring DL models. Computation of DL frameworks is accelerated by the GPU processing which rapidly improves computation. The complexity of CUDA programming is addressed by GPU processing recently since GPUs can gather multiple cores without extensive usage of resources or energy consumption. The research work highlights significance of the GPUs, multi-core systems and their benefits on different disciplines and contributes towards providing optimized approaches to

follow. The ease of using edge computing is a new possibility since most of the IoT devices these days have built-in GPUs. For future, hardware accelerators such as Tensor Processing Unit (TPU) and Global Analogic Programming Unit (GAPU) are being explored since GPUs are not well-suited for ML based models.

ACKNOWLEDGMENT

This work is supported by the Faculty Research Committee of the Faculty of Information Technology, Majan University College, Muscat, Oman.

REFERENCES

[1] S. Kato, K. Lakshmanan, and Y. Ishikawa, "TimeGraph: GPU Scheduling for Real-Time Multi-Tasking Environments".

[2] A. Pattnaik *et al.*, "Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities," in *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, Haifa Israel: ACM, Sep. 2016, pp. 31–44. doi: 10.1145/2967938.2967940.

[3] M. S. Shaikh, C. Hua, M. A. Jatoi, M. M. Ansari, and A. A. Qader, "Application of grey wolf optimisation algorithm in parameter calculation of overhead transmission line system," *IET Science Measure & Tech*, vol. 15, no. 2, pp. 218–231, Mar. 2021, doi: 10.1049/smt2.12023.

[4] M. S. Shaikh, S. Raj, M. Ikram, and W. Khan, "Parameters estimation of AC transmission line by an improved moth flame optimization method," *Journal of Electrical Systems and Inf Technol*, vol. 9, no. 1, p. 25, Dec. 2022, doi: 10.1186/s43067-022-00066-x.

[5] S. Baruah and N. Fisher, "The partitioned scheduling of sporadic real-time tasks on multiprocessor platforms," in *2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, 2005, pp. 346–353. doi: 10.1109/ICPPW.2005.83.

[6] W. Dally, S. Keckler, and D. Kirk, "Evolution of the Graphics Processing Unit (GPU)," *IEEE Micro*, vol. 41, pp. 42–51, Nov. 2021, doi: 10.1109/MM.2021.3113475.

[7] M. Madiajagan and S. S. Raj, "Chapter 1 - Parallel Computing, Graphics Processing Unit (GPU) and New Hardware for Deep Learning in Computational Intelligence Research," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, A. K. Sangaiah, Ed., Academic Press, 2019, pp. 1–15. doi: https://doi.org/10.1016/B978-0-12-816718-2.00008-7.

[8] C. Tan *et al.*, "ARENA: Asynchronous Reconfigurable Accelerator Ring to Enable Data-Centric Parallel Computing." arXiv, Apr. 19, 2021. Accessed: Jan. 15, 2024. [Online]. Available: http://arxiv.org/abs/2011.04931

[9] B. Tang, L. Chen, W. Sun, and Z. Lin, "Review of surface defect detection of steel products based on machine vision," *IET Image Processing*, vol. 17, no. 2, pp. 303–322, Feb. 2023, doi: 10.1049/ipr2.12647.

[10] M. W. Ashour, M. M. Abdulrazzaq, and M. Siddique, "Machine Vision Inspection of Steel Surface Using Combined Global and Local Features," in *ITNG 2023 20th International Conference on Information Technology-New Generations*, S. Latifi, Ed., Cham: Springer International Publishing, 2023, pp. 359–368.

[11] S. Mittal and S. Vaishay, "A survey of techniques for optimizing deep learning on GPUs," *Journal of Systems Architecture*, vol. 99, p. 101635, 2019, doi: https://doi.org/10.1016/j.sysarc.2019.101635.

[12] Y. Maksum *et al.*, "Computational Acceleration of Topology Optimization Using Parallel Computing and Machine Learning Methods – Analysis of Research Trends," *Journal of Industrial Information Integration*, vol. 28, p. 100352, 2022, doi: https://doi.org/10.1016/j.jii.2022.100352.

[13] G. Pratx and L. Xing, "GPU computing in medical physics: A review: GPU computing in medical physics," *Med. Phys.*, vol. 38, no. 5, pp. 2685–2697, May 2011, doi: 10.1118/1.3578605.

[14] A. Cano, "A survey on graphic processing unit computing for large-scale data mining," *WIREs Data Min & Knowl*, vol. 8, no. 1, p. e1232, Jan. 2018, doi: 10.1002/widm.1232.

[15] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, Boston, MA, USA: IEEE, Apr. 2009, pp. 163–174. doi: 10.1109/ISPASS.2009.4919648.

[16] D. Song, C. Zhang, Y. Zhu, and J. Liu, "A Low Cost Cross-Platform Video/Image Process Framework Empowers Heterogeneous Edge Application," in *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video*, in NOSSDAV '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 22–28. doi: 10.1145/3592473.3592566.

[17] N. Kumar and C. Arun, "A novel video compression model based on GPU virtualization with CUDA platform using bi-directional RNN," *International Journal of Information Technology*, Oct. 2023, doi: 10.1007/s41870-023-01456-8.

[18] S. Tian, J. Huber, K. Parasyris, B. Chapman, and J. Doerfert, "Direct GPU Compilation and Execution for Host Applications with OpenMP Parallelism," in *2022 IEEE/ACM Eighth Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC)*, Dallas, TX, USA: IEEE, Nov. 2022, pp. 43–51. doi: 10.1109/LLVM-HPC56686.2022.00010.

[19] M. Siddique, T. Ahmed, and M. S. Husain, "An Empirical Approach to Monitor the Flood-Prone Regions of North India Using Sentinel-1 Images," *AETiC*, vol. 6, no. 4, pp. 1–14, Oct. 2022, doi: 10.33166/AETiC.2022.04.001.

[20] Q. T. Pham, T. Q. Nguyen, P. C. Hoang, Q. H. Dang, D. M. Nguyen, and H. H. Nguyen, "A review of SNN implementation on FPGA," in *2021 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, 2021, pp. 1–6. doi: 10.1109/MAPR53640.2021.9585245.

[21] D. Li, S. Byna, and S. Chakradhar, "Energy-Aware Workload Consolidation on GPU," in *2011 40th International Conference on Parallel Processing Workshops*, Taipei City, Taiwan: IEEE, Sep. 2011, pp. 389–398. doi: 10.1109/ICPPW.2011.25.

[22] P. Czarnul, "Investigation of Parallel Data Processing Using Hybrid High Performance CPU," *cai*, vol. 39, no. 3, pp. 510–536, 2020, doi: 10.31577/cai_2020_3_510.

[23] S. Chen, Z. Li, B. Yang, and G. Rudolph, "Quantum-Inspired Hyper-Heuristics for Energy-Aware Scheduling on Heterogeneous Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1796–1810, 2016, doi: 10.1109/TPDS.2015.2462835.

[24] D. C. Kiran, S. Gurunarayanan, J. P. Misra, and A. Nawal, "Global Scheduling Heuristics for Multicore Architecture," *Scientific Programming*, vol. 2015, pp. 1–12, 2015, doi: 10.1155/2015/860891.

[25] M. D. Hill and M. R. Marty, "Amdahl's Law in the Multicore Era," *Computer*, vol. 41, no. 7, pp. 33–38, 2008, doi: 10.1109/MC.2008.209.

[26] D. H. Woo and H.-H. S. Lee, "Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era," *Computer*, vol. 41, no. 12, pp. 24–31, Dec. 2008, doi: 10.1109/MC.2008.494.

[27] K. Parasyris, G. Georgakoudis, E. Rangel, I. Laguna, and J. Doerfert, "Scalable Tuning of (OpenMP) GPU Applications via Kernel Record and Replay," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver CO USA: ACM, Nov. 2023, pp. 1–14. doi: 10.1145/3581784.3607098.

[28] M. Kawai, A. Ida, T. Hanawa, and K. Nakajima, "Dynamic Core Binding for Load Balancing of Applications Parallelized with MPI/OpenMP," in *Computational Science – ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023, Proceedings, Part III*, Berlin, Heidelberg: Springer-Verlag, 2023, pp. 378–394. doi: 10.1007/978-3-031-36024-4_30.

[29] O. Surakhi, M. Khanafseh, and S. Sarhan, “A Survey on Parallel Multicore Computing: Performance & Improvement,” *Adv. sci. technol. eng. syst. j.*, vol. 3, no. 3, pp. 152–160, Jun. 2018, doi: 10.25046/aj030321.

[30] J. M. R. Borbon, J. Huang, B. M. Wong, and W. Najjar, “Acceleration of Parallel-Blocked QR Decomposition of Tall-and-Skinny Matrices on FPGAs,” *ACM Trans. Archit. Code Optim.*, vol. 18, no. 3, May 2021, doi: 10.1145/3447775.

[31] M. Qasaimeh *et al.*, “Benchmarking vision kernels and neural network inference accelerators on embedded platforms,” *Journal of Systems Architecture*, vol. 113, p. 101896, Feb. 2021, doi: 10.1016/j.sysarc.2020.101896.

[32] J. Cong, Z. Fang, M. Lo, H. Wang, J. Xu, and S. Zhang, “Understanding Performance Differences of FPGAs and GPUs”.

[33] M. N. Waheed and M. Siddique, “Real-Time Scheduling Models in Diverse Multi-core OpenMP Applications,” in *2021 International Conference on Decision Aid Sciences and Application (DASA)*, 2021, pp. 702–705. doi: 10.1109/DASA53625.2021.9682396.