International Journal of Multidisciplinary Research and Publications
ISSN (Online): 2581-6187

GPU Concepts and Graph Application Challenges: A Review

Shilan Ahmed Mohammed[1], Rezgar Hasan Saeed[2], Jihan Abdulazeez Ahmed[3], Shilan Bashar Muhammad[4], Zainab Salih Ageed[5], Zryan Najat Rashid[6]
[1]IT Management, Dohuk Polytechnic University, Duhok-Iraq, Shilan.itm@gmail.com
[2]Computer Science, Near East University, Cyprus, rezgarhasan1992@gmail.com
[3]Computer Science, University of Duhok, Duhok-Iraq, drjihanrasool@uod.ac
[4]Mathmatic, University of Duhok, Duhok-Iraq, shilan.mohammed@uod.ac
[5]Translation, Nawroz University, Duhok-Iraq, zainab.ageed@nawroz.edu.krd
[6]Computer Network, Sulaimani Polytechnic University, Sulaimani-Iraq, zryan.rashid@spu.edu.iq

Abstract**—** *Due to the tremendous development that the world is witnessing day by day, this development included various sciences such as medicine, engineering, space, and exact sciences that require high-resolution visual representation and massive graphic processing. Many research and studies appeared on GPU. Many researchers and developers appeared who devoted Their time and effort to studying GPU and its importance and influential role in various fields. In this review paper, a brief synopsis of history, applications, challenges, and some research in various areas of life covers some aspects of GPU.*

Keywords**—** *GAS model, Graph processing, GPU, Graph, BSP model.*

I. INTRODUCTION

A graphics processing unit (PC / GPU) is a computer chip that renders graphics and images by performing rapid mathematical calculations [1]. GPUs are used for both continual technical and recreational computing. GPUs have traditionally been in charge of rendering 2D and 3D images, animations, and film. A GPU is a special circuit designed to easily access and modify the memory so that pictures can be produced in a frame buffer intended for transmission to a display [2]. GPUs are used on personal devices, mobile phones and social networks. [3]. Modern GPUS are highly effective in controlling computer graphics, personal computers, workstations and game consoles [4]. Their parallel configuration enables them to be more powerful than generic CPUs for algorithms with substantial parallel data processing [5]. The general-purpose graphics processing unit (GPU) is being used to change the stream processor, which is used as the new "left behind" stream processor [6]. The NVIDIA compute architecture is like a general-purpose computer in that you can direct it to perform several different tasks [7]. This idea of computing with modern graphics accelerators enables the use of these accelerators for general-purpose computing, in addition to performing graphical operations [8]. CUDA by NVIDIA gives a vital portion of a standard C program to run on the stream processors available to the hardware inside the primary GPU. This technology speeds up the use of dedicated hardware and opens up new types of applications [9]. This means that a classic C program can take advantage of one or more GPUs to rapidly run-on large matrices in parallel while also allowing it to switch from the CPU to the GPU when necessary [10]. CUD Accord API also enables CPU-based programs to directly access GPU capabilities for more general programming without the restriction of using graphics APIs (CUDA Application Program Interface) [11-16].

A graph mostly consists of a logical series of vertices and edges that link these pairs. These graphs have been implemented in recent real-world problem areas as linkages between objects [17]. Here are some examples of implementations of graphs and GPU in real life.

- Electronic Design Automation
- Bioinformatics
- Data Science, Analytics, and Databases
- Media and Entertainment
- Computational Fluid Dynamics
- Machine Learning
- Computational Finance
- Défense and Intelligence
- Imaging and Computer Visions
- Medical Imaging

Furthermore, graphs are also used by computer systems to show different structures. Also, graphs are frequently used in compilation optimization [18].

II. GPU CONCEPTS AND SIGNIFICANTS

A. GPU Architecture History and Evolution

The new graphic card was first produced in 1995 when the first 3D add-in was introduced. Three generations from the point of view of parallel architecture [19].

A —Fixed functional architecture: Each hardware device has graphic functions in this generation between 1995 and 2000. Both operations are performed on the stream representation on the processor cores. A lot of graphics processing can be supported by the use of the GPU [20].

B—Separated shader architecture: The GeForce 3 from NVIDIA brought programmable pixel shading to the consumer market in 2001 [21]. This helped significantly increase the expressiveness and versatility of fast graphics processing. They were usually used in games [22].

Shilan Ahmed Mohammed, Rezgar Hasan Saeed, Jihan Abdulazeez Ahmed, Shilan Bashar Muhammad, Zainab Salih Ageed, and Zryan Najat Rashid, "GPU Concepts and Graph Application Challenges: A Review," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 3, Issue 12, pp. 52-57, 2021.

C— Unified shader architecture: In 2012, NVIDIA launch Kepler technology that uses dynamic parallelism. The Pascal architecture was implemented for higher performance computing. In 2016, with the advancement of AI, Deep Learning, autonomous driving systems, and numerous other computer-intensive applications, NVIDIA released Pascal to enhance performance [23].

B. GPU Computing Architecture (Modern)

The image of a current GPU is complete. A GPU consists of several streaming processors, each containing a variety of GPU cores, special function units, registers, double accuracy units(s), and a thread scheduler [24]. Moreover, the unit with floating point arithmetic executes most of GPU program instructions. A GPU that enables 32 to 96 threads in existing hardwires is offered with multithreading [25]. A GPU memory can be divided into two memory hierarchies: on-device memory and on-chip memory. Dynamic Random Access is the on-device memory DRAM,

DRAM consists of local memory, global memory, continuous memory, and texture, while memory on the chip includes shared memory registers, cache L1/L2, and cache & texture constants [26]. A limited portion of CPU memory is accessible to any thread. The conceptual memory of constant and texture is built for graphical calculations directly. In order to achieve on-device memory, the average latencies usually need hundreds of clock cycles [27].

Additional cache types L1/L2. For stream computing GPUs are sometimes used and a cache is not needed. Each SM has a limited memory (48 KB per SM in the NVIDIA K40 GPU3), but high speed and only the threads on the SM are accessible [28]. (This is the main reason the contribution of the die-area to the cache in GPU is minimal.) GPUs are commonly used as a co-processor for the host CPU [29]. A PCI-Express cable links a GPU to the host. The data transfer (DMA) between the on-board GPU memory and the machine's main CPU memory [30]. The zero-copy feature is provided with the host and machine addresses for recent GPU architectures such as CUDA and OpenCL. This technique thus has a positive effect on communication performance [31].

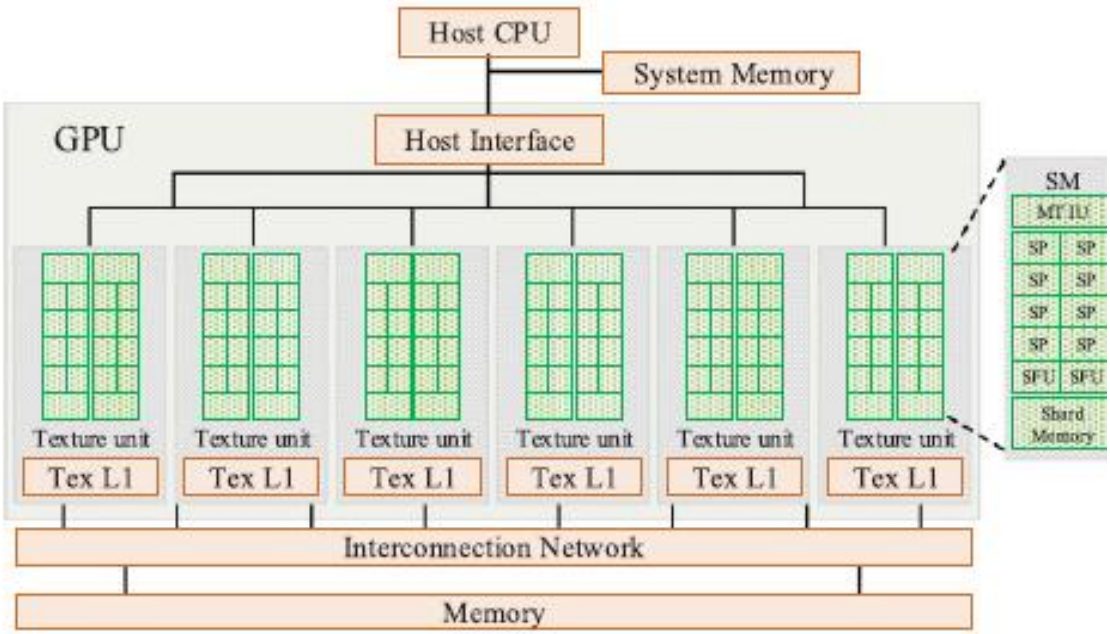

Fig. 1. GPU Architecture

C. GPU APIs and Programming Models

We add OpenGL, Direct3D, CUDA and OpenCL as GPU APIs and Program Models since remote API and other GPU virtualization approaches primarily aim at virtualizing the above mentioned libraries and models. [32].

OpenGL is a graphics acceleration library for accessing GPU hardware. The library specializes in the implementation of video games, image processing, and technical application visualization tasks [33]. OpenGL offers an application programming interface (API) that is hardware-independent and implemented on multiple applications [34].

Graphics cards, regardless of the device software underlying them [18].

Direct3D is a Microsoft Windows proprietary graphics API. Direct3D a Direct3D [35]. It is a low-level API for 3D graphics for performance-intensive games and others [36]. Direct3D provides coherent and general abstraction of complex GPU hardware implementations, which reveal advanced graphical features including z-buffering, w-buffering, stencil buffering and spatial counter-aliasing [37].

CUDA is a parallel programming paradigm developed by NVIDIA. It enables CUDA allowed GPUs to simulcast popular information in order to convert the graphic card by

Shilan Ahmed Mohammed, Rezgar Hasan Saeed, Jihan Abdulazeez Ahmed, Shilan Bashar Muhammad, Zainab Salih Ageed, and Zryan Najat Rashid, "GPU Concepts and Graph Application Challenges: A Review," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 3, Issue 12, pp. 52-57, 2021.

software developers [38]. Typically processing units for graphics (GPGPU). CUDA's programming languages C and C++ are similarly linked and have a limited number of primitive elements for device memory allocation, data transfer, GPU kernel execution, event management and nuclear sync and other languages operations. [39].

OpenCL is a specification for heterogeneous architectures for parallel applications. OpenCL defines the programming language of a C-like computer kernel, OpenCL. APIs can also run OpenCL kernels, which can be administered with memory transfer from the host to the computer. The only difference is that CUDA can only be performed by NVIDIA [5, 40].

D. Graph Programming Models

A vertex-centric programming interface supports programming simplicity much of the existing GPU graph processing platform [41]. They used the function is immediately iterated on each vertex. There are also two models for concurrent graph editing [42]. The two main terms are GAS and BSP (Bulk Synchronous Parallel).

1- Model GAS. GAS is an acceptable graphical algorithm programming model Vertex-centrically implemented, these three functions: collect, submit, and disperse are usually implemented. As such, the GAS model is divided into three phases [43].

The step of the meeting. In this step a vertex collects information from neighbouring vertices and edges using the user-defined gathering feature [44].

The application process, the user-defined application feature, is called a vertex depending on the data gathered during the collection phase. In this step Vertex values are modified as the application function is called. No interaction exists between vertices [45].

Scatter process phase. In this step, the new value (s) is spread to all adjacent vertices and edges. With updates pushed to isolated vertices, some implementations use the push-style scatter. This can also cause some push-style algorithms to ignore the gather process to reduce edge-traversals [46].

The GAS model, which simplifies the research method for the sophistication and correctness of the Graph algorithms applied using this model, summarize the overall synchronization [47].

But since the GPU sync overhead is not zero, we shouldn't overlook it when implementing a method that follows this paradigm for graph-processing. CUDA only supports, for example. Sync in the same block between threads. The computer divides computing into many kernels such that all threads in different blocks are globally cohesive [48].

Each kernel's endpoint serves as a global barrier because the GPU runs kernels one by one. It is expensive to use both local block synchronization and global syncing. An important job is to decrease the number of sync-points in graphs To minimize sync-points [49].

2- BSP Model.

The BSP model is implemented in a number of basic stages. The simultaneous procedures are carried out asynchronously during each super stage and communicate with each other by sending and receiving message [50]. Both processes are synchronised with the use of a barrier at the end of a super phase. The following three stages in each method comprise, in turn, a super step:

- Local computing: Computation functions are performed locally.
- Global communications occur in this phase.
- Synchronization of barriers: synchronization and synchronization of all computations and communication Guaranteed to be achieved at this point.

III. Graph Application Challenges

Any past study has been studied to address the complexities of graphics processing for GPUs.

—Mapping workload: the GPU is modeled as several threads with single instructions (SIMTs). Parallel graphical calculations also create load imbalance due to an unusual graph structure. The CPU has a powerful and versatile control device that can adjust the timetable as necessary [51].

Memory Access Pattern: Typically, the processor has a big main memory that is suitable for most graphs in the real world. The processing of large graphs is a difficulty because the GPU only has minimal memory [52]. This issue could be solved by applying partitioning methods to data tables. On how to organize irregular graph data, there is relatively sparse research.

—Data layout: As a graph processing technique to gain contiguous memory access. This is a global cache in a GPU that all GPUs share [53]. It is helpful to feed data to multiple SIMD threads. Graph algorithms and graph data structures afford erratic data access. In order to get the benefits of GPU, the level of parallelism is very limited for irregular data layouts [54].

— Various things The Processor is robust and can handle conditional algorithms for branching. Branch divergence happens as several threads in a state branch take opposite paths in the same wavefront. This causes serious problems in GPU efficiency because only one direction can be implemented at once in SIMD mode [55]. Threads are clustered into parallel GPU programming in blocks to facilitate interthread coordination and memory sharing. The kernel architecture significantly affects the degree of parallelism [56].

IV. Literature Review

In recent years, much research's concerning GPU has been studied. In this section, we discuss some of these research Z. Zheng, et al. [57] Proposed a coloring algorithm for a high performance graph that blends the recursive technique with the sequential method of spreading Felucca. In the first step, Felucca uses a recursive routine to color most vertexes in the diagram. The sequential spreading strategy is then used to colorize the remaining vertices to prevent clashes of the recursive algorithm.

The routine and effective simulation of large chemical and biological systems is being implemented with a modern heterogeneous CPU+GPU enhanced DFTB approach. I. Allen,

Shilan Ahmed Mohammed, Rezgar Hasan Saeed, Jihan Abdulazeez Ahmed, Shilan Bashar Muhammad, Zainab Salih Ageed, and Zryan Najat Rashid, "GPU Concepts and Graph Application Challenges: A Review," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 3, Issue 12, pp. 52-57, 2021.

et al. [58] A variety of specific implementations, hardware design benchmarks, and frameworks for this technical application have been examined for different broad range of chemical and biological systems in order to determine and understand the performance of this heterogeneous CPU+GPU approach. Finally, as an example of an incredibly large/complex structure, a broad-scale DFTB MD simulation of accurate HIV proteases (3,974 in total) is conducted to show implementation capacity and is, as you know, first treated in detail at the quantum-MD level of directly solved safety protection. The concept was introduced for high-performance LEO integrated GPUs by the UGA Small Satellite Research Laboratory.

C. Adams et al. [59] Combining an internal miniature GPU/SoC system with a standard flight computer. Such technology paves the way for a large number of NASA targets, including space-based AI, computer vision and neural networks.

Wu et al. [60] Using the Fluid-Imbiting Particle(FLIP) technique, using an accurate, large-scale simulation fluid on GPU hardware via the sparse grid hierarchy defined in NVIDIA GVDB Voxels. The system deals with tens of millions of particles in a virtually unbounded simulation domain, explains novel techniques for parallel, sparse grid hierarchy, and fast incremental updates for GPU transfer of particles. Furthermore, the FLIP solution includes a slim and work-efficient collection of parallel data from particle to voxel and a matrix-free GPU gradient solver for sparse grids.

Djenou et al. [61] Make the most of the parallel GPU in bees swarm optimization There is no equivalent progress in the on-board data processing and downlink technologies for mining massive datasets, by improving the GBSO-miner in earth monitoring (EO).

L. Davidson et al. [62] The new GPU has suggested an accelerated system of onboard data processing and a built-in program to prove feasible throughput processing and compressor performance. By taking an analogy between heat processes and mass transportation processes in the urban dispersion model.

Kristóf et al. [63] Using graphics processing unit based software originally intended for the application of mechanical engineering. The software enables the geometry to be modified and the temporary flow and concentration fields to be visualized during the simulation so that various design principles can be studied and compared.

Zhu et al. [64] It suggested a new algorithm in the time series results that identifies all related sequences. Using CUDA's GPGPU and developing a methodology to optimize memory access for the implementation of GPUs. Pagoda, a runtime framework that virtualizes the GPU resources presented by Tsung Tai Yeh et al. [65] using an OS-like daemon kernel called the MasterKernel.

SiGAMMA (Server-based integrated GPU Arbitration System for Memory Access) was used by N. Capodieci et al. [66] to remove interference with CPU tasks caused by overlapping GPU memory requests.

A parallel CCD algorithm introduced by Du et al. [67] aims to accelerate the removal of N-body CCD by spreading the load through a high-performance GPU cluster

V. Comparison and Discussion

GPU is used in different fields, and several researchers have worked on it; we mention the following

- Graph Coloring
- Chemical and Biological Systems.
- For Small Satellites as a Flight Computer.
- First, a spatially sparse, complete FLIP solver runs entirely on the GPU.
- reduce the interference between CPU and GPU tasks by SiGAMMA
- To improve bee's swarm
- NASA and Space application
- Urban Dispersion Studies
- Discovery Method in Time Series Data
- for Narrow Tasks, And many other fields.

Table 1: some research on GPU

Ref.	field	Used	Result
[57]	Graph Coloring	Felucca Algorithm	improve the graph coloring achieve 1.19-8.39 speedup over the state-of-the-art algorithms.
[58]	extensive chemical and biological systems	new heterogeneous CPU+GPU	high computational performance
[59]	Space	for Small Satellites as a Flight Computer	to meet many of NASA's goals that require space-based AI
[60]	Fast Fluid Simulations	FLIP technique	Faster than running on the CPU.
[61]	bees swarm optimization	GBSO-Miner	up to 800 times faster than an optimized CPU Implementation
[62]	Space application	low power embedded GPU	Reduce data corruption from up to 46% to 2%, execution time overhead of 130%.
[63]	Urban Dispersion Studies	Large Eddy Simulation (LES)	The model results show a satisfactory agreement
[64]	find all similar subsequences in the time-series data	Dynamic Time Warping (DTW) algorithm	achieve better performance with using GPU parallelization
[65]	Narrow Tasks	Pagoda	achieves a speedup of 5.70x, 1.51x, 1.69x over PThreads, CUDA-HyperQ, GeMTC Respectively and much lower latency per task
[66]	eliminating the interference on CPU and GPU tasks	SiGAMMA	effective to solve inflation
[67]	Parallel Continuous Collision	parallel CCD algorithm	more computationally efficient than existing sequential CCD approaches.

Shilan Ahmed Mohammed, Rezgar Hasan Saeed, Jihan Abdulazeez Ahmed, Shilan Bashar Muhammad, Zainab Salih Ageed, and Zryan Najat Rashid, "GPU Concepts and Graph Application Challenges: A Review," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, Volume 3, Issue 12, pp. 52-57, 2021.

VI. CONCLUSIONS

In this paper, a brief review of the kinds of literature related to the GPU is presented in different fields like Chemical, biological, space, and some narrow tasks. This research offers some information on GPU's concepts, significance, applications, challenges, and graph processing model. It can give information to someone who is new and does not have enough information on GPU.

REFERENCES

[1] B. T. Jijo, S. R. Zeebaree, R. R. Zebari, M. A. Sadeeq, A. B. Sallow, S. Mohsin, *et al.*, "A comprehensive survey of 5G mm-wave technology design challenges," *Asian Journal of Research in Computer Science*, pp. 1-20, 2021.

[2] L. M. Abdulrahman, S. R. Zeebaree, S. F. Kak, M. A. Sadeeq, A.-Z. Adel, B. W. Salim, *et al.*, "A state of art for smart gateways issues and modification," *Asian Journal of Research in Computer Science*, pp. 1-13, 2021.

[3] X. Shi, X. Luo, J. Liang, P. Zhao, S. Di, B. He, *et al.*, "Frog: Asynchronous graph processing on GPU with hybrid coloring model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 29-42, 2017.

[4] H. Shukur, S. R. Zeebaree, A. J. Ahmed, R. R. Zebari, O. Ahmed, B. S. A. Tahir, *et al.*, "A State of Art Survey for Concurrent Computation and Clustering of Parallel Computing for Distributed Systems," *Journal of Applied Science and Technology Trends*, vol. 1, pp. 148-154, 2020.

[5] Z. S. Ageed, S. R. Zeebaree, M. A. Sadeeq, M. B. Abdulrazzaq, B. W. Salim, A. A. Salih, *et al.*, "A state of art survey for intelligent energy monitoring systems," *Asian Journal of Research in Computer Science*, pp. 46-61, 2021.

[6] H. R. Abdulqadir, S. R. Zeebaree, H. M. Shukur, M. M. Sadeeq, B. W. Salim, A. A. Salih, *et al.*, "A study of moving from cloud computing to fog computing," *Qubahan Academic Journal*, vol. 1, pp. 60-70, 2021.

[7] Z. S. Ageed, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, Z. N. Rashid, A. A. Salih, *et al.*, "A survey of data mining implementation in smart city applications," *Qubahan Academic Journal*, vol. 1, pp. 91-99, 2021.

[8] B. Goodarzi, F. Khorasani, V. Sarkar, and D. Goswami, "High Performance Multilevel Graph Partitioning on GPU," in *2019 International Conference on High Performance Computing & Simulation (HPCS)*, 2019, pp. 769-778.

[9] F. Q. Kareem, S. R. Zeebaree, H. I. Dino, M. A. Sadeeq, Z. N. Rashid, D. A. Hasan, *et al.*, "A survey of optical fiber communications: challenges and processing time influences," *Asian Journal of Research in Computer Science*, pp. 48-58, 2021.

[10] A. B. Sallow, M. Sadeeq, R. R. Zebari, M. B. Abdulrazzaq, M. R. Mahmood, H. M. Shukur, *et al.*, "An Investigation for Mobile Malware Behavioral and Detection Techniques Based on Android Platform," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 22, pp. 14-20.

[11] J. R. Cheng and M. Gen, "Accelerating genetic algorithms with GPU computing: A selective overview," *Computers & Industrial Engineering*, vol. 128, pp. 514-525, 2019.

[12] P. Yang, F. Dong, V. Codreanu, D. Williams, J. B. Roerdink, B. Liu, *et al.*, "Improving utility of GPU in accelerating industrial applications with user-centered automatic code translation," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 1347-1360, 2017.

[13] A. R. Arsadjaja and A. I. Kistijantoro, "Online Speech Decoding Optimization Strategy with Viterbi Algorithm on GPU," in *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, 2018, pp. 130-134.

[14] H. Lee, M. Shafique, and M. A. Al Faruque, "Aging-aware workload management on embedded GPU under process variation," *IEEE Transactions on Computers*, vol. 67, pp. 920-933, 2018.

[15] T. Ben-Nun, M. Sutton, S. Pai, and K. Pingali, "Groute: An asynchronous multi-GPU programming model for irregular computations," *ACM SIGPLAN Notices*, vol. 52, pp. 235-248, 2017.

[16] X. Yi, J. Duan, and C. Wu, "Gpunfv: a gpu-accelerated nfv system," in *Proceedings of the First Asia-Pacific Workshop on Networking*, 2017, pp. 85-91.

[17] M. A. Sulaiman, M. Sadeeq, A. S. Abdulraheem, and A. I. Abdulla, "Analyzation Study for Gamification Examination Fields," *Technol. Rep. Kansai Univ*, vol. 62, pp. 2319-2328, 2020.

[18] X. Shi, Z. Zheng, Y. Zhou, H. Jin, L. He, B. Liu, *et al.*, "Graph processing on GPUs: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, pp. 1-35, 2018.

[19] Z. Ageed, M. R. Mahmood, M. Sadeeq, M. B. Abdulrazzaq, and H. Dino, "Cloud computing resources impacts on heavy-load parallel processing approaches," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 22, pp. 30-41, 2020.

[20] O. Alzakholi, H. Shukur, R. Zebari, S. Abas, and M. Sadeeq, "Comparison among cloud technologies and cloud performance," *Journal of Applied Science and Technology Trends*, vol. 1, pp. 40-47, 2020.

[21] Z. S. Ageed, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, H. S. Yahia, M. R. Mahmood, *et al.*, "Comprehensive survey of big data mining approaches in cloud systems," *Qubahan Academic Journal*, vol. 1, pp. 29-38, 2021.

[22] R. J. Hassan, S. R. Zeebaree, S. Y. Ameen, S. F. Kak, M. A. Sadeeq, Z. S. Ageed, *et al.*, "State of Art Survey for IoT Effects on Smart City Technology: Challenges, Opportunities, and Solutions," *Asian Journal of Research in Computer Science*, pp. 32-48, 2021.

[23] M. A. Omer, S. R. Zeebaree, M. A. Sadeeq, B. W. Salim, S. x Mohsin, Z. N. Rashid, *et al.*, "Efficiency of malware detection in android system: A survey," *Asian Journal of Research in Computer Science*, pp. 59-69, 2021.

[24] M. A. Sadeeq and S. Zeebaree, "Energy management for internet of things via distributed systems," *Journal of Applied Science and Technology Trends*, vol. 2, pp. 59-71, 2021.

[25] A. A. Salih, S. R. Zeebaree, A. S. Abdulraheem, R. R. Zebari, M. A. Sadeeq, and O. M. Ahmed, "Evolution of Mobile Wireless Communication to 5G Revolution," *Technology Reports of Kansai University*, vol. 62, pp. 2139-2151, 2020.

[26] A. A. Yazdeen, S. R. Zeebaree, M. M. Sadeeq, S. F. Kak, O. M. Ahmed, and R. R. Zebari, "FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review," *Qubahan Academic Journal*, vol. 1, pp. 8-16, 2021.

[27] A. S. Abdulraheem, A. A. Salih, A. I. Abdulla, M. A. Sadeeq, N. O. Salim, H. Abdullah, *et al.*, "Home automation system based on IoT," 2020.

[28] M. Sadeeq, A. I. Abdulla, A. S. Abdulraheem, and Z. S. Ageed, "Impact of Electronic Commerce on Enterprise Business," *Technol. Rep. Kansai Univ*, vol. 62, pp. 2365-2378, 2020.

[29] A. I. Abdulla, A. S. Abdulraheem, A. A. Salih, M. A. Sadeeq, A. J. Ahmed, B. M. Ferzor, *et al.*, "Internet of Things and Smart Home Security," *Technol. Rep. Kansai Univ*, vol. 62, pp. 2465-2476, 2020.

[30] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, "IoT and Cloud computing issues, challenges and opportunities: A review," *Qubahan Academic Journal*, vol. 1, pp. 1-7, 2021.

[31] S. M. S. A. Abdullah, S. Y. A. Ameen, M. A. Sadeeq, and S. Zeebaree, "Multimodal emotion recognition using deep learning," *Journal of Applied Science and Technology Trends*, vol. 2, pp. 52-58, 2021.

[32] S. Zeebaree, S. Ameen, and M. Sadeeq, "Social media networks security threats, risks and recommendation: A case study in the kurdistan region," *International Journal of Innovation, Creativity and Change*, vol. 13, pp. 349-365, 2020.

[33] D. H. Maulud, S. R. Zeebaree, K. Jacksi, M. A. M. Sadeeq, and K. H. Sharif, "State of art for semantic analysis of natural language processing," *Qubahan Academic Journal*, vol. 1, pp. 21-28, 2021.

[34] I. M. Ibrahim, "Task scheduling algorithms in cloud computing: A review," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, pp. 1041-1053, 2021.

[35] Z. S. Ageed, R. K. Ibrahim, and M. A. Sadeeq, "Unified Ontology Implementation of Cloud Computing for Distributed Systems," *Current Journal of Applied Science and Technology*, pp. 82-97, 2020.

[36] A. Sallow, S. Zeebaree, R. Zebari, M. Mahmood, M. Abdulrazzaq, and M. Sadeeq, "Vaccine tracker," *SMS reminder system: Design and implementation*, 2020.

[37] H. S. Yahia, S. R. Zeebaree, M. A. Sadeeq, N. O. Salim, S. F. Kak, A.-Z. Adel, *et al.*, "Comprehensive Survey for Cloud Computing Based

Nature-Inspired Algorithms Optimization Scheduling," *Asian Journal of Research in Computer Science*, pp. 1-16, 2021.
[38] S. Zeebaree and H. M. Yasin, "Arduino based remote controlling for home: power saving, security and protection," *International Journal of Scientific & Engineering Research*, vol. 5, pp. 266-272, 2014.
[39] H. M. Yasin, S. R. Zeebaree, and I. M. Zebari, "Arduino Based Automatic Irrigation System: Monitoring and SMS Controlling," in *2019 4th Scientific International Conference Najaf (SICN)*, 2019, pp. 109-114.
[40] I. M. Zebari, S. R. Zeebaree, and H. M. Yasin, "Real Time Video Streaming From Multi-Source Using Client-Server for Video Distribution," in *2019 4th Scientific International Conference Najaf (SICN)*, 2019, pp. 109-114.
[41] S. Zeebaree and I. Zebari, "Multilevel Client/Server Peer-to-Peer Video Broadcasting System," *International Journal of Scientific & Engineering Research*, vol. 5, 2014.
[42] H. Malallah, S. R. Zeebaree, R. R. Zebari, M. A. Sadeeq, Z. S. Ageed, I. M. Ibrahim, *et al.*, "A Comprehensive Study of Kernel (Issues and Concepts) in Different Operating Systems," *Asian Journal of Research in Computer Science*, pp. 16-31, 2021.
[43] H. M. Yasin, S. R. Zeebaree, M. A. Sadeeq, S. Y. Ameen, I. M. Ibrahim, R. R. Zebari, *et al.*, "IoT and ICT based Smart Water Management, Monitoring and Controlling System: A Review," *Asian Journal of Research in Computer Science*, pp. 42-56, 2021.
[44] H. I. Dino, S. R. Zeebaree, A. A. Salih, R. R. Zebari, Z. S. Ageed, H. M. Shukur, *et al.*, "Impact of Process Execution and Physical Memory-Spaces on OS Performance."
[45] H. H. A. Razaq, A. S. Gaser, M. A. Mohammed, E. T. Yassen, S. A. Mostafad, S. R. Zeebaree, *et al.*, "Designing and Implementing an Arabic Programming Language for Teaching Pupils," *Journal of Southwest Jiaotong University*, vol. 54, 2019.
[46] B. S. Osanaiye, A. R. Ahmad, S. A. Mostafa, M. A. Mohammed, H. Mahdin, R. Subhi, *et al.*, "Network Data Analyser and Support Vector Machine for Network Intrusion Detection of Attack Type."
[47] S. Zeebaree, N. Cavus, and D. Zebari, "Digital Logic Circuits Reduction: A Binary Decision Diagram Based Approach," *LAP LAMBERT Academic Publishing*, 2016.
[48] Z. A. Younis, A. M. Abdulazeez, S. R. Zeebaree, R. R. Zebari, and D. Q. Zeebaree, "Mobile Ad Hoc Network in Disaster Area Network Scenario: A Review on Routing Protocols," *International Journal of Online & Biomedical Engineering*, vol. 17, 2021.
[49] S. A. Mostafa, A. Mustapha, A. A. Ramli, R. Darman, S. R. Zeebaree, M. A. Mohammed, *et al.*, "Applying Trajectory Tracking and Positioning Techniques for Real-time Autonomous Flight Performance Assessment of UAV Systems," *Journal of Southwest Jiaotong University*, vol. 54, 2019.
[50] S. R. Zeebaree, O. Ahmed, and K. Obid, "CSAERNet: An Efficient Deep Learning Architecture for Image Classification," in *2020 3rd International Conference on Engineering Technology and its Applications (IICETA)*, 2020, pp. 122-127.
[51] S. M. Mohammed, K. Jacksi, and S. Zeebaree, "A state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, pp. 552-562, 2021.
[52] A. M. Abdulazeez, S. R. Zeebaree, and M. A. Sadeeq, "Design and Implementation of Electronic Student Affairs System," *Academic Journal of Nawroz University*, vol. 7, pp. 66-73, 2018.
[53] M. A. Sadeeq, S. R. Zeebaree, R. Qashi, S. H. Ahmed, and K. Jacksi, "Internet of Things security: a survey," in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 162-166.
[54] S. R. Zeebaree and H. Rajab, "Design and Implement a Proposed Multi-Sources to Multi-Destinations Broadcasting Video-Signals," in *2019 4th Scientific International Conference Najaf (SICN)*, 2019, pp. 103-108.
[55] D. Zebari, H. Haron, and S. Zeebaree, "Security issues in DNA based on data Hiding: A review," *International Journal of Applied Engineering Research*, vol. 12, pp. 0973-4562, 2017.
[56] I. A. Khalifa, S. R. Zeebaree, M. Ataş, and F. M. Khalifa, "Image steganalysis in frequency domain using co-occurrence matrix and Bpnn," *Science Journal of University of Zakho*, vol. 7, pp. 27-32, 2019.
[57] Z. Zheng, X. Shi, L. He, H. Jin, S. Wei, H. Dai, *et al.*, "Feluca: A Two-Stage Graph Coloring Algorithm With Color-Centric Paradigm on GPU," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, pp. 160-173, 2020.
[58] S. I. Allec, Y. Sun, J. Sun, C.-e. A. Chang, and B. M. Wong, "Heterogeneous CPU+ GPU-enabled simulations for DFTB molecular dynamics of large chemical and biological systems," *Journal of chemical theory and computation*, vol. 15, pp. 2807-2815, 2019.
[59] C. Adams, A. Spain, J. Parker, M. Hevert, J. Roach, and D. Cotten, "Towards an integrated GPU accelerated SoC as a flight computer for small satellites," in *2019 IEEE Aerospace Conference*, 2019, pp. 1-7.
[60] K. Wu, N. Truong, C. Yuksel, and R. Hoetzlein, "Fast fluid simulations with sparse volumes on the GPU," in *Computer Graphics Forum*, 2018, pp. 157-167.
[61] Y. Djenouri, D. Djenouri, A. Belhadi, P. Fournier-Viger, J. C.-W. Lin, and A. Bendjoudi, "Exploiting GPU parallelism in improving bees swarm optimization for mining big transactional databases," *Information Sciences*, vol. 496, pp. 326-342, 2019.
[62] R. L. Davidson and C. P. Bridges, "Error resilient GPU accelerated image processing for space applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, pp. 1990-2003, 2018.
[63] G. Kristóf and B. Papp, "Application of GPU-based Large Eddy Simulation in urban dispersion studies," *Atmosphere*, vol. 9, p. 442, 2018.
[64] H. Zhu, Z. Gu, H. Zhao, K. Chen, C.-T. Li, and L. He, "Developing a pattern discovery method in time series data and its GPU acceleration," *Big Data Mining and Analytics*, vol. 1, pp. 266-283, 2018.
[65] T. T. Yeh, A. Sabne, P. Sakdhnagool, R. Eigenmann, and T. G. Rogers, "Pagoda: Fine-grained GPU resource virtualization for narrow tasks," *ACM SIGPLAN Notices*, vol. 52, pp. 221-234, 2017.
[66] N. Capodieci, R. Cavicchioli, P. Valente, and M. Bertogna, "Sigamma: Server based integrated gpu arbitration mechanism for memory accesses," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, 2017, pp. 48-57.
[67] P. Du, E. S. Liu, and T. Suzumura, "Parallel continuous collision detection for high-performance GPU cluster," in *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2017, pp. 1-7.