GPU Shader Analysis and Power Optimization Model

Guruprasad Konnurmath
School of Computer Science and Engineering, KLE Technological University, India
guruprasad.konnurmath@kletech.ac.in (corresponding author)

Satyadhyan Chickerur
Centre for High Performance Computing, KLE Technological University, India
chickerursr@kletech.ac.in

Received: 2 December 2023 | Revised: 18 December 2023 | Accepted: 21 December 2023

 DOI: https://doi.org/10.48084/etasr.6695

ABSTRACT

With the rapid advancements in 3D game technology, workload characterization has become crucial for each new generation of games. The increased complexity of scenes in 3D games allows for stunning real-time visual quality. However, handling such workloads results in significant power consumption over the GPU rendering pipeline. The focus of the current paper is low power optimization, targeting texture memory, geometry engine, pixel, and rasterization, as these components are significant contributors to the power consumption of a typical GPU. The proposed methodology integrates the Dynamic Voltage Frequency Scaling (DVFS) technique, adjusting voltage and frequency based on the workload analysis of frame rates with respect to the scenes of 3D games. Frame rates of 60 fps and 30 fps are set up to understand and manage the workload on frames. Furthermore, for comparative analysis, various frame-level power analysis schemes such as No DVFS implemented, Frame History Method, Frame Signature Method, and Tiled History-based are introduced. The proposed scheme consistently surpasses these frame-level schemes, with fewer missed deadlines, while having the lowest energy consumption per frame rate. The implementation resulted in a remarkable 65% improvement in quality, indicated by a reduction in deadline misses, along with a substantial 60% energy saving.

Keywords-3D scene rendering pipeline; frames; GPU; DVFS; geometry; pixel; texture; workload

I. INTRODUCTION

In the current landscape of multimedia devices and applications, 3D models have gained widespread usage, particularly in high-definition video games, which constitute to be a major part of computer graphics. In the latest Top 500 supercomputer rankings, 101 of these supercomputers are powered by GPUs for acceleration [1]. There are many 3D model applications that exhibit compelling variations in the workload in the sequence of frames. Some high intensity workloads of 3D models [2] do not suite graphics design to achieve better performance and time. The goal is to identify opportunities for optimizing power consumption without compromising visual quality and user experience. An algorithm based on Dynamic Voltage Frequency Scaling (DVFS) [3, 4], is implemented to explore the characteristics confined to frame controlled rendering GPUs and reach desirable power savings. The workload of each frame is analyzed in the 3D scene by considering factors such as geometry [5, 6], textures, pixels [7], and rasterization. This analysis helps in understanding computational intensity of each frame. Matrix computing operations are commonly regarded as the cornerstone of various computations in the field of scientific applications.

As the digital world continues to expand, its environmental impact, sustainability, cost savings, scalability, battery life, and heat dissipation are major power consumption issues that have to be addressed. The experimental findings in similar works [8, 9] show decrease in energy consumed by GPUs up to 20%. The trade-offs between performance and power consumption in GPUs involve a delicate balance, and designers must make strategic decisions to optimize both aspects. Higher clock frequencies generally lead to better performance. But higher clock frequencies often require higher voltage levels, resulting in increased power consumption [3, 4, 10]. To enhance both speed and the quality of rendering, while considering the significance of textures and geometry, cutting-edge, high-performance texture streaming systems have been developed [11, 12]. The power consumption in shaders is influenced by a combination of factors related to the complexity of the shader code, the number of shader invocations, and memory access patterns. Geometric complexity increases the number of vertices and brings a series of operations in the vertex shader with transformations and lighting calculations, leading to higher power consumption. High-resolution textures or frequent sampling can increase memory and bandwidth usage, contributing to higher power consumption. Sophisticated visual effects also contribute towards more power consumption.

These data are utilized for predicting the workload for the frames [13]. At the onset of the decoding process, buffering is applied to each frame comprising of GoP (Group of Pictures) and the workload estimation is calculated for the buffered frames. Minimal value relating frequency-voltage during the decoded period for meeting the deadlines for GoPs is noted in [14]. Figure 1 exhibits the varying processing time of successive frames in the following games: Cyberpunk 2077, Assassin's Creed Valhalla, Cow Boy Simulator, Tomb Raider, Forsaken World. It is observed from the graph that the workload on the frames 10-15 is much lower compared with the workload of frames in the range of 1-7. Considering this major gap with respect to workload variations, considerable power savings can be achieved by adjusting the computational capability of GPUs based on workload variation in frames.

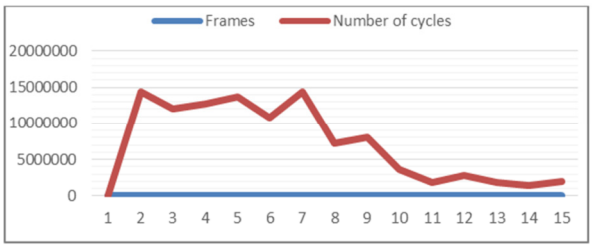

Fig. 1. Workload variation observed in the games.

II. METHODOLOGY

The architectural design of the proposed methodology is based on the input of the 3D scene specific to game applications and is depicted in Figure 2. The implemented architecture primarily comprises four key components:

- Video encoding and decoding for the frame rate analysis of the scenes.
- Workload characterization on parameter geometry, pixel, texture, and raster.
- Graphics Rendering Parameters Control (GRPC), which serves as a functionality that changes the parameters to influence the power consumed during runtime.
- The DVFS technique is responsible for managing the core frequencies of GPU.

In the initial stage, GPU's power consumption and connected resources used under rendering of the 3D scene are examined. To achieve continuous parallelism and placement on the device, the libraries of TensorFlow are used. Power consuming parameters include: geometric complexity, texture memory, pixelization, and textures workload. The key strategies include DVFS and frame rate analysis based on workload. The computation of workload considers the parameters for every frame rendered in a 3D game scene. Utilizing these frame workloads as a basis, DVFS method is implemented, involving adjustments to both GPU voltage and frequency. The frequency threshold is established ranging from a minimum of 30 to a maximum of 60 frames per second (fps). If the current framerate is below the target framerate, the frequency is augmented. Conversely, if the current framerate surpasses the target, the frequency is diminished to attain optimal power savings. Examining the present load status of the GPU, which furnishes details regarding its idle intervals and workload distribution, enables the reduction of power consumption through the fine-tuning of frequency-voltage variations on the dedicated GPUs.

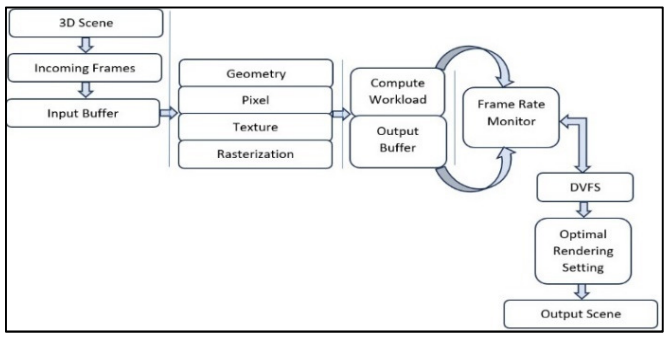

Fig. 2. Architecture of the proposed methodology.

The workload of a scene from a 3D model refers to the computational effort or processing requirements needed to render and display that scene on a computer or GPU. Finally, after careful analysis of the workloads observed on each frame, four parameters, namely geometry workload, pixel workload, texture workload, and rasterization workload contribute in possessing the workload for each frame of the rendered scene along with their major contribution in power consumption. The geometry workload on scenes refers to the computational effort and processing requirements related to the geometric aspects of a 3D scene during rendering. The total count in vertex shader instructions (Wv) under each frame can be calculated by:

$$Wv = V \times Nv \quad (1)$$

where Nv is the total length of instructions for each vertex shader program, and V represents the vertices total count for each frame. The geometry stage workload is mainly determined by the number of vertices processed by the shader program multiplied by the instruction length. The equation for Geometry workload (Wg) as a combination of workload on the vertex shader (Wv) and the workload observed due to clipping and binning, which is considered to be proportional to the count of primitives (P) is defined as:

$$Wg = Wv + P = V \times Nv + P \quad (2)$$

Pixel shading is a fundamental aspect of determining the final appearance of objects in the rendered image. The measure of workload identified due to pixel shading (Wp) is expressed as:

$$Wp = A \times Np \quad (3)$$

where Np depicts the pixel shader program length, indicating the amount of computations required for each pixel. The texture workload measure Wt can be estimated by considering the texels count that are read for every pixel under examination. The workload on texture can be expressed by:

$$Wt = A \times M \times t \quad (4)$$

where A represents the texels count on each pixel, capturing the workload due to basic texture access, M represents the number of textures associated with each pixel, considering scenarios where multiple textures are used for blending or multi-texturing effects, and t represents the number of texel samples required for filtering. Enabling early z-test in our experiments reveals that around 50% of the pixels are ignored at this particular stage, leading to a significant reduction in both pixel shading and texturing workloads. Hence, with early z enabled:

$$Wp = 0.5 \times A \times Np \text{ and } Wt = 0.5 \times A \times M \times t \quad (5)$$

Operations on rasters, including depth-stencil testing and color blending, primarily contribute to frame buffer memory's bandwidth. For R raster operations in each primitive, every pixel for primitives require $2 \times R$ rate of access towards buffers of frame. The respective operations are included with the sum of reading and writing from and to the memory of the frame buffer, hence the factor of 2 is considered. Hence, raster operations workload (Wr) is characterized by:

$$Wr = A \times 2 \times R \quad (6)$$

During workload analysis for the parameter texturing, rasterization and pixel shading operations are executed parallelly, so the processing speed is determined through the slowest parameter executed. Hence, correlative increment in the workload is found from the increment in any one of these parameters. From Figure 3, we see that the proposed methodology has almost ignorable under prediction count compared to the frame oriented signature-based scheme. Function tf.debugging.set_log_device_placement checks out the device placement. The tf.function and input_pipeline_analyzer display descriptive GPU core analysis results, including GPU's step-time and duration of GPU time spent. TensorFlow Stats exhibits the performance for each TensorFlow op, executed on the device. Nvidia management tool NVML is employed for monitoring the power consumption in the graphics card.

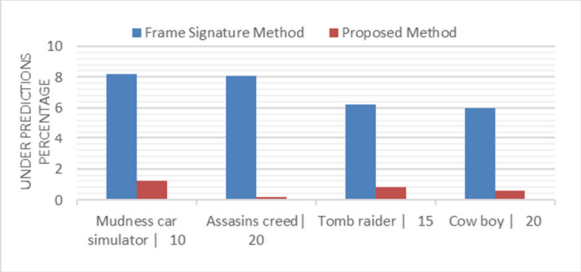

Fig. 3. Accuracy of the proposed method predictions.

III. EXPERIMENTAL SETUP AND IMPLEMENTATION

NVIDIA Nsight [15] is a powerful profiling tool providing in-depth analysis of GPU performance and frame rates. The frames of six 3D games are considered: Cyberpunk (CP) provides an excellent benchmark for testing frame rates and performance on high-end systems. Assassin's Creed Valhalla (AC) depicts the workload characterized on each frame. Cow Boy simulator (CB) possess intense action sequences, making it ideal for frame analysis. Tomb Raider (TR) features complex lighting, shadows, and detailed textures that stress the rendering pipeline. Forsaken World (FW) and Mudness offroad Car simulator (MC) are great for analyzing GPU performance under heavy rendering loads. Deadline misses, achieved frame rates, and normalized energy consumption at 30 and 60 fps are analyzed for various frame level power analysis schemes. In the suggested approach, we assess frame correlation by comparing the geometric workload of the current frame to a previous frame. If the workloads of the frames are similar, we consider previous frame's workload as a reliable estimate of the existing current frame. In our experiments, we set empirically the threshold for geometry workload to 10%. After completing the geometry processing, we analyzed other three frame components to determine the optimal processing frequency. Based on the workload analysis, the frames are divided into two categories:

- Light Frames: Frames with an estimated workload that is equal to or less than previous frame's workload.

$$W(fi) <= W(fi-1) \quad (8)$$

Suppose the current frame rate surpasses 60 fps. Within this range we systematically decrease the core frequency of the GPU. This sequence ensures that the workload is effectively managed with the minimal consumption of resources and power. Consequently, if the frame rate drops below the lower limit of 30 fps, the GPU core's frequency level is increased to enhance the frame rate.

- Heavy Frames: Frames with an anticipated workload that surpasses previous frame's workload:

$$W(fi) > W(fi-1) \quad (9)$$

In the second scenario, we focus on the workload of the present frame. To handle this augmented workload, we operate at the maximum available frequency.

The frame workload Algorithm using DVFS is presented below.

Algorithm 1: Proposed frame workload-based DVFS method

Step 1: Initialization: Set initial values for fps and corresponding voltage and frequency levels

Min =30 fps, *Max* = 60 fps and standard voltage and frequency levels

Step 2: Frame and workload analysis of the current frame. Calculate geometry workload, pixel workload, raster operations workload, and texture workload from (2), (3), (4), and (6).

Step 3: Frame workload classification: $low \leq 30$ fps, *medium* = 45 fps, *high* >= 60 fps.

Step 4: Voltage and Frequency Adjustment: If the current frame workload <= previous frame workload decrease frequency levels. Else increase frequency levels.

Step 5: Frame processing: Process current frame using adjusted V and f levels to conserve power.

Step 6: Performance evaluation: Compare the achieved performance with the desired target set. Assess whether the performance is meeting the desired criteria or further adjustments is needed.

Step 7: Repeat steps 1 to 6 for subsequent frames in the sequence

Step 8: End the algorithm when all the frames have been processed.

IV. RESULTS AND DISCUSSION

For experimentation, the results are generated for GPU processors compatible with 3, 6, 9, and 12 levels of frequency and voltage, providing options in adjustments over DVFS. Results are produced for frame rates with thresholds of 30 fps and 60 fps, enabling us to evaluate performance and efficiency in two contexts: (i) slack utilization for lighter frame rates and (ii) guaranteeing the capability to attain deadlines for higher frame rates. In Figure 4, we can observe comparative percentages of deadline misses for the No DVFS implemented (WD) method, Frame History Method (FHM), Frame Signature Method (FSM), Tiled History-based Method (THM), and our Proposed Method (PM), for the CP game at a target frame rate of 60 fps. Notably, the FHM DVFS scheme exhibits the highest number of deadlines misses. The FSM performs better compared to FHM but still results in notable deadline misses. The PM excels due to its ability to implement early corrective measures, leading to reduction in the number of under-predictions. Comparing 3-level and 2-level DVFS schemes, it is observed that the 3-level scheme results into deadline misses. The 2-level scheme opts *Fmax* frequency to prevent deadline misses, but this decision might lead to some slack under-utilization. The proposed methodology in turn exhibits superior efficiency in handling both scenarios effectively.

Figure 5 illustrates the impact of deadline misses. In the proposed methodology we can reach a frame rate that closely reflects the predictive frame rate, meanwhile the frame-based DVFS technique undergoes notable decrease in the frame rate. Referring towards Table I, history-based schemes lead to a drop of up to 10 fps, but the PM experiences maximum drop around 1 fps. Most often, required predictive frequency F underlies among two levels, $F1$ and $F2$ ($F1 < F < F2$). As a result, the frame is processed at $F2$ to meet the timing constraints. Similarly, in the History and Signature Method schemes, the frame with the average frequency nearer to F can be reached alternating between frequencies $F1$ and $F2$ for time durations $t1$ and $t2$ such that:

$$F1 \times t1 + F2 \times t2 = F \times TD, t1 + t2 = TD$$

In our experiments, we found that using this approach resulted in more than 40% of the frames missing their deadlines.

However, in the proposed DVFS scheme, we can effectively utilize the slack obtained from processing a frame at $F2$ to slow down future frames, allowing us to process the frame at an average frequency closer to the desired F. This is observed through the results achieved with respect to frames at 60 fps, as in Figure 6. But, for the processed frames at 30 fps, as in Figure 7, we conclude that the PM consumes a bit more power compared to the legacy history-based schemes. To attain proper understanding of the comparison with respect to power efficiency introduced in each scheme, Normalized Energy per Normalized Frame Rate is evaluated. From the results examined in Figure 8, it is observed that the Energy per Frame Rate is minimized under the implemented frame-based DVFS scheme which outperforms frame history and signature DVFS techniques in terms of energy efficiency. The results for the remaining applications and various frame rate per second requirements, at 60 fps, are provided in Figures 9-11.

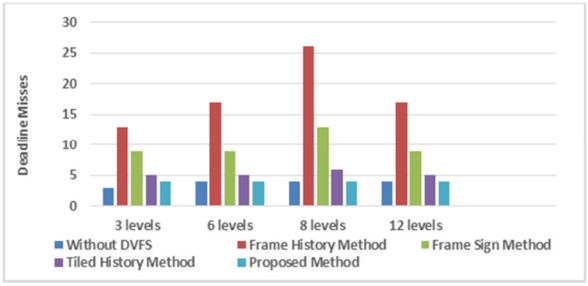

Fig. 4. Deadline misses at the desired frame rate of 60 fps.

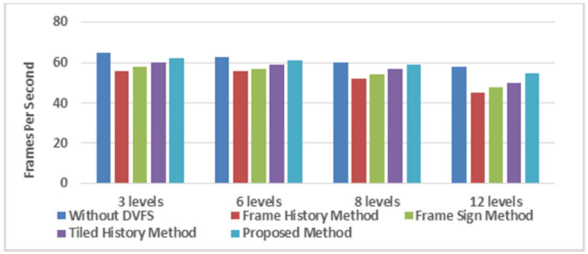

Fig. 5. Achieved frame rate at the desired target of 60 fps.

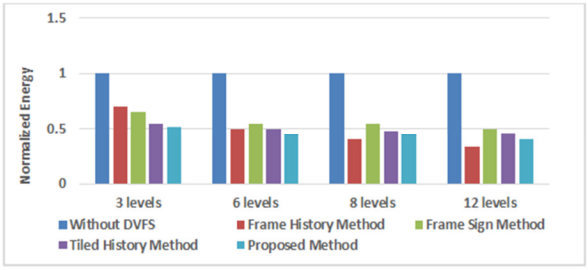

Fig. 6. Normalized energy consumption at a frame rate of 60 fps.

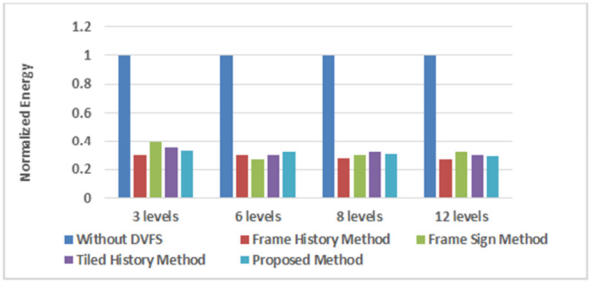

Fig. 7. Normalized energy consumption at a frame rate of 30 fps.

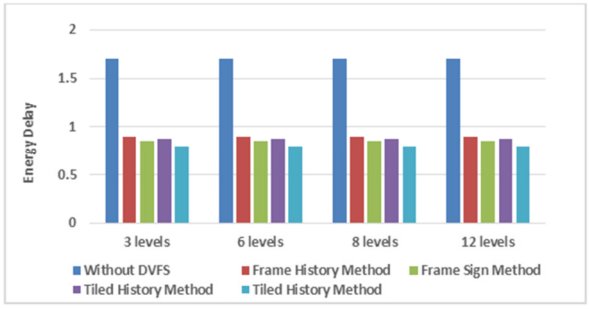

Fig. 8. Energy consumption per frame rate when operating at 30 fps.

Deadline Misses

	(3 Levels)							(6 Levels)					
	CP	AC	CB	TR	FW	MC		CP	AC	CB	TR	FW	MC
WD	0	1	2	3	4	4.8	WD	0	1	2	3	4	4.8
FHM	10	8	10	11	10	10.5	FHM	13	9	15	17	18	18.5
FSM	7	8	9	10	11	11.6	FSM	8	8	12	12	13	13.5
THM	2	3	6	8	10	10.5	THM	2	4	6	8	9	11
PM	0	2	3	3.8	4	5	PM	0	2	3	4	4	5
	(9 Levels)							(12 Levels)					
WD	0	1	2	3	4	4.8	WD	0	1	2	3.2	4.5	5
FHM	20	17	19	22	23	25	FHM	21	18	19.3	22.4	24	27
FSM	10	11	12	14	14	15	FSM	11	11	13	15	16	18
THM	2	7	8	11	13	10	THM	3	8	9	12	14	12
PM	0	2	3	4	4	5	PM	0	3	4	4.5	5	5.7

Fig. 9. DVFS results at 60 fps, regarding deadline misses.

Frame Rate

	(3 Levels)							(6 Levels)					
	CP	AC	CB	TR	FW	MC		CP	AC	CB	TR	FW	MC
WD	50	54	55	56	58	59	WD	50	54	55	56	58	59
FHM	42	45	42	47	50	52	FHM	40	42	40	45	48	50
FSM	45	45	46	47	48	49	FSM	45	45	45.5	46	46	47
THM	48	50	48	48	49	49.5	THM	48	50	48	48	49	49.5
PM	50	51	51.5	52	52.4	53	PM	50	51	51.5	52	52.4	53
(9 Levels)							(12 Levels)						
WD	50	54	55	56	58	59	WD	50	54	55	56	58	59
FHM	36	40	46	22	23	21	FHM	36	40	46	22	23	21
FSM	40	43	42	42	43	42	FSM	40	43	42	42	43	42
THM	46	48	49	49	50	50	THM	46	48	49	49	50	50
PM	48	48	49	49	51	51	PM	48	48	49	49	51	51

Fig. 10. DVFS results at 60 fps, regarding framerates.

V. CONCLUSION

Following a thorough examination, we identified the vertex processing, texture operations, geometry engine, and rasterization as significant factors influencing the power consumption of a typical GPU. The comparative outcomes of the proposed approach, in relation to both history-based and signature-based methods, demonstrate notable power conservation.

Nomalized Energy/ Frame Rate

	(3 Levels)							(6 Levels)					
	CP	AC	CB	TR	FW	MC		CP	AC	CB	TR	FW	MC
WD	1.00	1.00	1.02	1.02	1.03	1.03	WD	1.00	1.00	1.02	1.02	1.03	1.03
FHM	0.50	1.00	1.00	1.02	1.03	1.03	FHM	0.45	1.00	1.03	1.03	1.04	1.04
FSM	0.30	1.01	1.01	1.02	1.02	1.03	FSM	0.42	1.03	1.05	1.06	1.06	1.08
THM	0.28	0.80	0.82	0.90	0.92	0.95	THM	0.43	0.90	0.92	0.95	0.95	0.96
PM	0.20	0.72	0.76	0.78	0.83	0.90	PM	50	51	51.5	52	52.4	53
	(9 Levels)							(12 Levels)					
WD	1.00	1.00	1.02	1.02	1.03	1.03	WD	1.00	1.00	1.02	1.02	1.03	1.03
FHM	0.42	0.90	1.01	1.01	1.02	1.02	FHM	0.40	0.90	1.01	1.02	1.03	1.03
FSM	0.35	0.80	0.90	0.92	0.95	0.96	FSM	0.35	0.80	0.90	0.92	0.95	0.96
THM	0.38	0.78	0.88	0.90	0.92	0.92	THM	0.38	0.80	0.90	0.92	0.93	0.94
PM	0.30	0.70	0.80	0.82	0.85	0.88	PM	0.33	0.74	0.85	0.87	0.87	0.89

Fig. 11. DVFS results at 60 fps, focusing on normalized energy consumption per frame rate.

By examining the available literature and contrasting with older games, it becomes evident that contemporary games are driven by the quest for enhanced realism, achieved by incorporating more intricate geometry. This shift in paradigm has resulted in a significant rise in the demand on vertex processing. Hence, we introduced low-power optimizations directed at these components. Our implementation demonstrated a substantial 65% improvement in quality, as indicated by a reduction in deadline misses compared to alternative history-based DVFS schemes. Moreover, it resulted in noteworthy energy conservation of 60%. In conclusion, it is clearly understood that graphics applications display workload variations and as a result, low-power optimization techniques have been introduced specifically designed for shader components. Future work includes the use of machine learning algorithms for predicting power consumption in shaders. By training models on historical data and runtime parameters, it may be possible to create more accurate predictions and guide dynamic optimizations.

REFERENCES

[1] "Home - | TOP500." https://top500.org/.

[2] R. Li, A. Arora, S. Li, Q. Wu, and L. K. John, "Hardware-aware 3D Model Workload Selection and Characterization for Graphics and ML Applications," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, Apr. 2022, https://doi.org/10.1109/ISQED54688.2022.9806296.

[3] A. Mishra and N. Khare, "Analysis of DVFS Techniques for Improving the GPU Energy Efficiency," *Open Journal of Energy Efficiency*, vol. 4, no. 4, pp. 77–86, Nov. 2015, https://doi.org/10.4236/ojee.2015.44009.

[4] J. Guerreiro, A. Ilic, N. Roma, and P. Tomás, "DVFS-aware application classification to improve GPGPUs energy efficiency," *Parallel Computing*, vol. 83, pp. 93–117, Apr. 2019, https://doi.org/10.1016/j.parco.2018.02.001.

[5] N. C. Kundur, B. C. Anil, P. M. Dhulavvagol, R. Ganiger, and B. Ramadoss, "Pneumonia Detection in Chest X-Rays using Transfer Learning and TPUs," *Engineering, Technology & Applied Science Research*, vol. 13, no. 5, pp. 11878–11883, Oct. 2023, https://doi.org/10.48084/etasr.6335.

[6] P. D. Chung, "Smoothing the Power Output of a Wind Turbine Group with a Compensation Strategy of Power Variation," *Engineering,*

Technology & Applied Science Research, vol. 11, no. 4, pp. 7343–7348, Aug. 2021, https://doi.org/10.48084/etasr.4234.

[7] J. R. Monfort and M. Grossman, "Scaling of 3D game engine workloads on modern multi-GPU systems," in *Proceedings of the Conference on High Performance Graphics 2009*, New York, NY, USA, May 2009, pp. 37–46, https://doi.org/10.1145/1572769.1572776.

[8] G. Konnurmath and S. Chickerur, "Power-Aware Characteristics of Matrix Operations on Multicores," *Applied Artificial Intelligence*, vol. 35, no. 15, pp. 2102–2123, Dec. 2021, https://doi.org/10.1080/08839514.2021.1999013.

[9] G. Konnurmath and S. Chickerur, "An Investigation into Power Aware Aspects of Rendering 3D Models on Multi-Core Processors," *Procedia Computer Science*, vol. 218, pp. 887–898, Jan. 2023, https://doi.org/10.1016/j.procs.2023.01.069.

[10] T. P. Minh *et al.*, "Finite Element Modeling of Shunt Reactors Used in High Voltage Power Systems," *Engineering, Technology & Applied Science Research*, vol. 11, no. 4, pp. 7411–7416, Aug. 2021, https://doi.org/10.48084/etasr.4271.

[11] A. Zhang, K. Chen, H. Johan, and M. Erdt, "High-performance adaptive texture streaming and rendering of large 3D cities," *The Visual Computer*, vol. 38, no. 4, pp. 1245–1262, Apr. 2022, https://doi.org/10.1007/s00371-021-02152-z.

[12] D. Geisler, I. Yoon, A. Kabra, H. He, Y. Sanders, and A. Sampson, "Geometry types for graphics programming," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, Aug. 2020, Art. np./173, https://doi.org/10.1145/3428241.

[13] A. Mackin, F. Zhang, and D. R. Bull, "A Study of High Frame Rate Video Formats," *IEEE Transactions on Multimedia*, vol. 21, no. 6, pp. 1499–1512, Jun. 2019, https://doi.org/10.1109/TMM.2018.2880603.

[14] N. Karpinsky and S. Zhang, "Holovideo: Real-time 3D range video encoding and decoding on GPU," *Optics and Lasers in Engineering*, vol. 50, no. 2, pp. 280–286, Feb. 2012, https://doi.org/10.1016/j.optlaseng.2011.08.002.

[15] "NVIDIA Nsight Systems," *NVIDIA Developer*. https://developer.nvidia.com/nsight-systems.