JRAEE

Journal of Robotics, Automation, and Electronics Engineering
Vol. 2, No. 1, 2024, pp. 39-50
e-ISSN 3025-3780
https://journal.uny.ac.id/publications/jraee

DVFS and Timing Optimization on GPU for Data Center Computation

Faris Yusuf Baktiar[a,1,*]

[a] Universitas Negeri Yogyakarta
[1] farisyusufbaktiar@uny.ac.id;
[*] Corresponding Author

ARTICLE INFO

Article History
Received 8 Feeb. 2024
Revised 28 Mar. 2024
Accepted 13 Jun. 2024

Keywords
Data Center;
GPU;
Computation;
DVFS;
Timing Optimization.

ABSTRACT

Data center computing requires efficient GPU support, both in terms of functionality and power consumption. GPU performance efficiency can be reduced due to high power usage and reduced GPU work stability. So it requires an analysis of computational performance and power efficiency to improve performance and reduce power usage. Core voltage, core frequency, and memory timings are parameters that affect the efficiency of computing performance, power efficiency, and stability. Increasing computational efficiency and GPU power with the effect of modifying parameters can be done through the Basic Input-Output System (BIOS). This study analyzes the efficiency of computational performance by optimizing memory timings and analyzing power efficiency and stability by modifying the DVFS algorithm. Tests are carried out using computational benchmarks commonly used in data centers including the tessellation algorithm, rendering, image processing, pi calculation, image stitching, deep learning, molecular simulation, and N-body. The efficiency of computing performance and GPU power efficiency can be increased by optimizing memory timings and changing the voltage and frequency values on DVFS. Increased performance efficiency ranged from 33.3% to 66.7% and power efficiency increased from 19.9% to 32.6%. Modification of the DVFS voltage state can increase voltage stability and GPU core frequency stability.

Komputasi pada Data center membutuhkan dukungan GPU yang efisien, baik dari segi kienrja maupu konsumsi daya. Efisiensi kinerja GPU dapat menurun dikarenkan penggunaan daya yang tinggi dan penurunan kestabilan kerja GPU. sehingga dibutuhkan analisis efisiensi kinerja komputasi dan daya untuk meningkatkan kinerja dan menurunkan penggunaan daya. Tegangan inti, frekuensi inti, dan timing memori merupakan parameter yang berpengaruh pada efisiensi kinerja komputasi, efisiensi daya, dan kestabilan. Peningkatan efisiensi komputasi dan daya GPU dengan modifikasi parameter yang berpengaruh dapat dilakukan melalui Basic Input-Output System (BIOS). Penelitian ini melakukan analisis terhadap efisiensi kinerja komputasi dengen optimasi timing memori, analisis efisiensi daya dan kestabilan dengan modifikasi algoritma DVFS. Pengujian dilakukan dengan menggunakan benchmark yang berisi komputasi yang biasa digunakan pada data center diantaranya algoritma tessellation, render, pemrosesan citra, kalkulasi pi, image stitching, deep learning, simulasi molekul, dan N-body.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

Komputasi pada Data center membutuhkan dukungan GPU yang efisien, baik dari segi kienrja maupu konsumsi daya. Efisiensi kinerja GPU dapat menurun dikarenkan penggunaan daya yang tinggi dan penurunan kestabilan kerja GPU. sehingga dibutuhkan analisis efisiensi kinerja komputasi dan daya untuk meningkatkan kinerja dan menurunkan penggunaan daya. Tegangan inti, frekuensi inti, dan timing memori merupakan parameter yang berpengaruh pada efisiensi kinerja komputasi, efisiensi daya, dan kestabilan. Peningkatan efisiensi komputasi dan daya GPU dengan modifikasi parameter yang berpengaruh dapat dilakukan melalui Basic Input-Output System (BIOS). Penelitian ini melakukan analisis terhadap efisiensi kinerja komputasi dengen optimasi timing memori, analisis efisiensi daya dan kestabilan dengan modifikasi algoritma DVFS. Pengujian dilakukan dengan menggunakan benchmark yang berisi komputasi yang biasa digunakan pada data center diantaranya algoritma tessellation, render, pemrosesan citra, kalkulasi pi, image stitching, deep learning, simulasi molekul, dan N-body. Efisiensi kinerja komputasi dan efisiensi daya GPU dapat ditingkatkan dengan optimasi timing memori dan pengubahan nilai tegangan dan frekuensi pada DVFS. Peningkatan efisiensi kinerja berkisar 33,3% hingga 66,7% dan efisiensi daya meningkat 19,9% hingga 32,6%. Modifikasi state tegangan DVFS dapat meningkatkan kestabilan tegangan dan kestabilan frekuensi inti GPU.

1. Introduction

The data center was once considered only as a large place for data collection and just a place for data traffic. Data centers are now changing from providing storage only to providing computing services. Science and engineering computing is currently being done in Data centers, such as Deep learning, molecular simulation, image processing, and physical calculation simulation. Increased computational burden on the data center, requires a device that can speed up the computing process in the Data center. GPU is one of the accelerator solutions available today for a Data center [1].

The computing paradigm for the application of science and engineering has now shifted. The Central Processing Unit (CPU) which used to be the computational backbone for the application of science and engineering, has now been assisted or even replaced by a Graphic Processing Unit (GPU). The CPU is considered no longer able to perform all computing tasks quickly and efficiently, especially for science and engineering applications. This is supported by the presence of GPGPU [2].

GPU which is used in data centers, has considerable power requirements. This is due to the number of core processors used on GPUs numbering hundreds to thousands. The impact of the high power requirements is the emergence of high temperatures on the GPU chip. High power can cause a decrease in GPU computing performance efficiency [3]. In addition to decreasing performance, temperature, and high power consumption, it can disrupt the stability of the work of the GPU while handling computing [4]. Disturbed stability can cause data to be damaged error, or lost.

Data centers require GPUs that have fast performance, are stable at high computational durations that are long enough, but still have relatively low power to avoid high temperatures caused by power dissipation. One alternative to reduce power usage is by lowering the GPU voltage, decreasing the GPU core frequency, and optimizing memory timings.

This can be done by using one of the settings in the GPU kernel [5], but this method still has challenges where there are often voltage or frequency drops that still do not work according to the configuration given. So that it often disrupts work stability. Then a voltage modification method,

timing, and GPU working frequency are needed. For the GPU to produce more stable, power-efficient performance, it can reduce GPU working temperature in computational intensive, such as its use in the data center.

The Basic Input-Output System (BIOS) is used in hardware as part of initialization. All states on the hardware will be initialized when the hardware boots. The BIOS has more access parameters than the kernel. This includes state voltage, voltage value, frequency value on DVFS, memory timings, and work frequency. Power efficiency and stability can be increased by changing the DVFS state and voltage values. The efficiency of computational performance can be improved by optimizing memory timing.

Seeing the problem of GPU computing performance efficiency in the data center which can be reduced by high power consumption and reduced GPU stability [6], this study will analyze the efficiency of computing performance and GPU power by timing optimization memory, changing the DVFS algorithm to the core voltage state, core voltage values and core frequency values embedded in the BIOS. This study focuses on analyzing the efficiency of computing performance, power efficiency, and stability of performance. The efficiency of computational performance, power efficiency, and stability are expected to be improved for computing using the GPU in the data center.

2. Method

This section explains how to implement BIOS optimization via DVFS and timing optimization. This section also explains the hardware and software used in this research as well as test methods that demonstrate computing stability and performance.

2.1 Overall Research Methods

The efficiency of GPU computing performance in this study will be improved by maintaining performance together by reducing GPU power consumption while increasing GPU stability. This is done to improve the efficiency of computing performance along with increasing the efficiency of power consumption with stable performance. Power efficiency will be improved by modifying the DVFS algorithm, both on state voltage, voltage value, and core frequency values. Computing performance will be maintained by optimizing memory timings. GPU computing performance can be measured by providing a benchmark containing the computational load that is processed to produce a measure of performance. The depiction of GPU computing performance in the data center requires a benchmark that contains computing commonly used in data centers such as pi digit calculations, 3D rendering and tessellation, molecular simulations, n-body simulations, and digital image processing. This research will use several algorithms that represent each of these computations. Computational benchmarks will be assisted by benchmarking software GPU power requirements in the computational process can be measured by recording GPU Power draw data when running computational benchmarks. Stability can be measured by recording core voltage and core frequency data from VRM and GPU cores.

2.2 Hardware Spesification Used

The hardware system that will be used in this study has the following specifications :

a. CPU AMD Ryzen 9 7900X 12 cores 24 threads running on 4.7 GHz
b. Motherboard AMD B650 Platform
c. RAM 64 GB (16GB X 4) DDR5 4800 MHz
d. Two GPUs AMD RX 6800 16GB GDDR6
e. Operating System: Windows 10 Pro 64-bit.

2.3 DVFS Optimization

The voltage and frequency at the GPU core are regulated using a power management algorithm on a computer component that is popular today, DVFS. DVFS works by adjusting the frequency and voltage at the GPU core to the given computational load. This is done so that the core frequency and voltage can adapt to the given load conditions to increase power efficiency [7].

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

DFVS in a BIOS is embedded in the GPU in the form of a voltage-state state setting. This state of voltage and frequency is a reference for the GPU kernel to carry out voltage and frequency adaptations. Reducing state frequency and voltage is done through the BIOS, by writing the state and value that you want to implement on the GPU.

Changing the DVFS rule is done at the voltage state. DVFS changes are expected to increase power efficiency and stability which will impact on improving the efficiency of computing performance [8]. Unstable voltage can be caused by the use of a voltage state where each state is too close. The voltage at the core is the main parameter in maintaining the frequency that runs at the core. The voltage state influences the power used and the frequency that can run at the GPU core. State voltage will be changed to less. GPU on default DVFS applies eight state voltages. State voltage will be converted into three voltage states, to accommodate idle state, mid-power state, and high-performance state

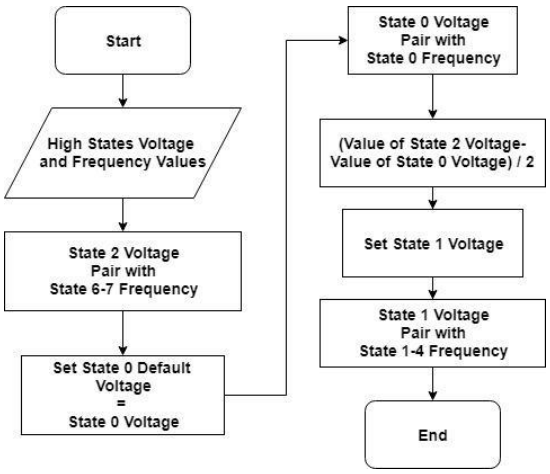

Fig. 1. DVFS modification flowchart

Many data center computations make the GPU work in two states, namely idle state, when not accepting computational loads, and high performance states when given high computational loads. This reduction in voltage state is expected to increase core and voltage frequency stability when given high computational loads to improve performance efficiency and reduce input power. The flow of changes in the voltage state of the DVFS is shown in Figure 1

2.4 Voltage and Frequency High-Performance State Tuning

Modification The state voltage in the DVFS algorithm requires a fixed voltage value for each frequency pair. Voltage states 1 to 7 on default DVFS have a variable value, only the state voltage 0 has a fixed value. Three voltage values are needed to meet the needs of three DVFS voltage states. **A**ssigning values to v0 will use the value v0 on the default DVFS. V2 value or upper voltage state will be taken based on the highest voltage reading from the kernel and then undervoled by 10-15%. This limit is given because the safe undervolting limit for computing use is 15-20%.

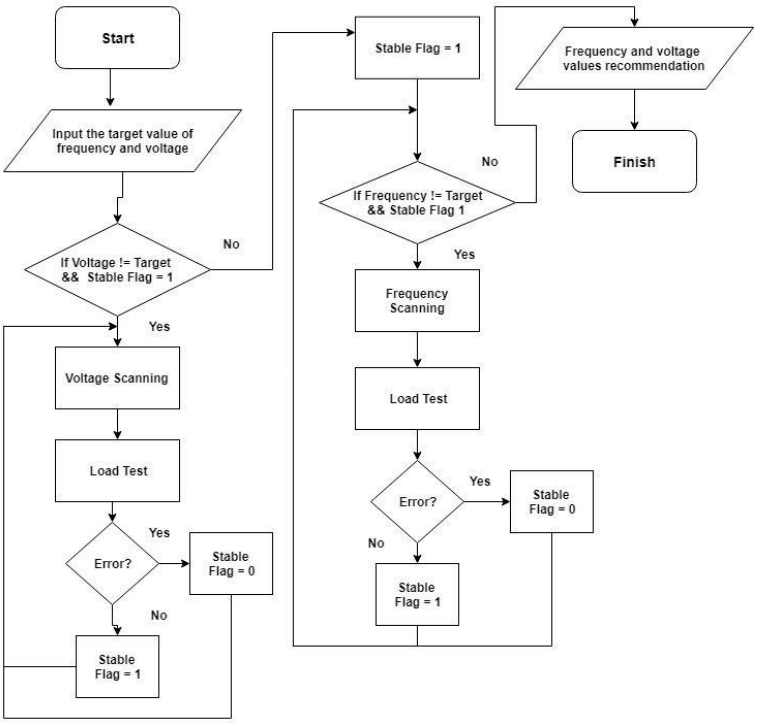

Fig. 2. Frequency tuning flowchart

Tuning is done on the kernel by scanning automatically through software, so it can be determined how much the undervolt can be done which will then be written on DVFS. The value v1 is the result of two from the difference between v0 and v2. Undervolting is done to produce lower voltage input at the GPU core so that the core power output can be lowered The upper frequency limit will be lowered to accommodate undervolting processes at the upper voltage state. Frequency tuning will be done on the kernel by automatic scanning through software with a 5% reduction limit. How it works tuning the upper limit of frequency and voltage using the scanning seen in the flowchart of Fig. 2.

The application of voltage and frequency values to be used at the highest state or state three will be determined by the tuning method. The other state, namely state one will follow the default voltage state zero in the default BIOS, and state two will be the difference between state one and three divided by two. Tuning is done by adjusting the frequency according to the voltage that want to achieve. The tuning process flowchart is shown in Fig. 2. The lowered voltage will result in the need for a decrease in frequency, so this study provides a limit of a decrease in the maximum frequency of 5% with a reduction in voltage targeted at 10% to 15%.

2.5 Memory Timing Optimization

Memory timing in the BIOS is in the form of a microcode as shown on the right side of **F**ig**.** 2. The microcode contains configuration timings at each work frequency level. The higher the frequency that is run, the timings used will be looser. The looser the memory timing, the slower the performance. Higher frequency states certainly have more loose timings, as seen in Figure 3. Timing optimization is done by utilizing each state'**s** timing. Microcode Timing at lower frequency states will be included at higher frequency states. This is intended to get high frequencies with tight timings. Increased memory computation perf**or**mance with a small power increase [9].

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

TIMING (MHz)	VALUE
250	1110000000000000022CC1C00628C110B205709080DC3600100204200210114
600	3330000000000000022CC1C00A520241940570B0B16C55103002264003A0514
900	3330000000000000022CC1C00E730362580570B0F9D860205002485005A0914
1000	3330000000000000022CC1C0008B5362990570B101FC7920500448600620A14
1125	3330000000000000022CC1C0029BD472FA0570C11234853060046A6006A0C14
1250	3330000000000000022CC1C004AC54834B0570C12A68803070046A700720E14
1375	3330000000000000022CC1C008CCD593AC0570D13AA09B4070048C7007A001
1500	5550000000000000022CC1C00AD515A3EC0570E142D4A64080048C700030114
1625	5550000000000000022CC1C00CE596B44D0570F1531CB2409004AE7000B0314
1750	7770000000000000022CC1C0010626C49D0571016B50BD509004AE700140514
1900	7770000000000000022CC1C00106A7D4FE0571117B98CA50A004C07011C0714
2000	7770000000000000022CC1C0031EE7D53F05711183BCD350B004C0701240814

Fig. 3. Memory timing microcode

2.6 Stability Test and Computation Performance Test

Stability testing is done by providing a computational load on the GPU. Computation given as a stability test is compute-intensive and memory-intensive. Compute intensive will burden the GPU core with the intensity of the use of high GPU cores and intensive memory computing will burden memory with computing that requires high memory allocation. Computing that meets compute-intensive and also memory-intensive one of them is 3D image rendering, Tessellation 3D computing, and Deep Learning Model [10]. Computing that is run will be facilitated using software to provide the computations tested.

The stability test was used also to see the effect of changing the state voltage on the DVFS method. Reducing the voltage state is expected to increase stability by being able to adapt voltage based on 3-state voltage and reduce voltage and frequency drop when given a high computational load. Benchmarking is done to see the performance magnitude of a computing device. Benchmarking will be used to see how much GPU computing performance. Benchmarking will be done by providing computing based on computing in the data center. Tests are carried out using computational benchmarks commonly used in data centers including the tessellation algorithm, rendering, image processing, pi calculation, image stitching, deep learning, molecular simulation, and N-body. Benchmarking software for each computing load will also acquire GPU computing performance data.

3. Result and Discussion

3.1 DVFS and Timing Optimization Result

The DVFS modification produces three state voltages and eight state DVFS frequencies, and memory timing optimization by tightening 2 steps tighter. The best upper limit of the DVFS voltage produced is the lowest voltage that runs with the upper limit of the highest core frequency.

The modified DVFS results in three state voltages with a value of each that is read in the BIOS reader of 800 mV, 900 mV, and 975 mV shown in Fig. 4, with readable values of sensors namely 825 mV, 925 mV, and 1025 mV. The upper limit of the core frequency is 1210 MHz down by 50MHz or 4% of the default frequency value. A decrease in the upper limit of the core voltage read by the sensor in the log data is 150mV or decreases by 12.8% from the upper limit of the default voltage. Memory timing optimization is 2 steps higher without increasing memory voltage.

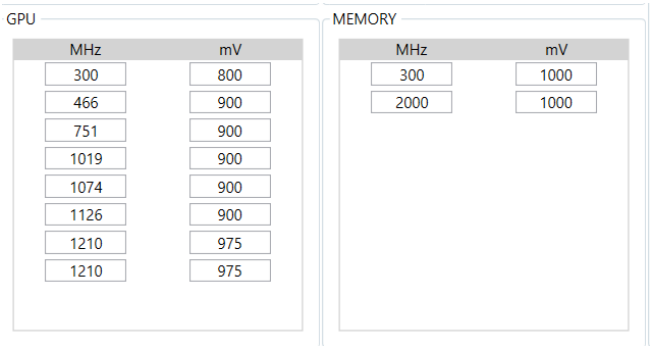

Fig. 4. State reduction and core frequency tuning result

3.2 Stability Test Analysis

Table 1 shows the modification of DVFS capable of increasing the level of stability compared to the default condition. Percentage percentages increased by 1.6% in 3D rendering computing. The difference is fairly insignificant but can describe the DVFS modification can achieve the same level of stability or even more than the default condition. DVFS modification also shows stable performance by being able to pass the test of stability in 3D tessellation computing.

Table 1. **Table 1.** Stability test result

3D Render Stability Test		3D Tessellation Stability Test	
Condition	Result	Condition	Result
Default	98.1%	Default	Pass
Modified	99.7%	Modified	Pass

The difference in the results of the stability of the default condition and after modification of DVFS in the 3D rendering computation test was due to a small difference in the stability of the framerate produced during the testing process. The 3D rendering stability test works by providing the workload on the GPU in the form of moving animated image frames, where the output is a series of images. Stability is measured by changes in frame rate. A decrease in framerate will reduce the level of stability. The decrease in framerate is caused by a decrease in core frequency as the main working parameter of the stability test. Further analysis of the effect of DVFS modification will be elaborated on the results of core voltage and frequency readings.

3.3 GPU core voltage stability analysis

The difference in the results of the stability of the default condition and after modification of DVFS in the 3D rendering computation test was due to a small difference in the stability of the framerate produced during the testing process. The 3D rendering stability test works by providing the workload on the GPU in the form of moving animated image frames, where the output is a series of images. Stability is measured by changes in frame rate. A decrease in framerate will reduce the level of stability. The decrease in framerate is caused by a decrease in core frequency as the main working parameter of the stability test. Further analysis of the effect of DVFS modification will be elaborated on the results of core voltage and frequency readings.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

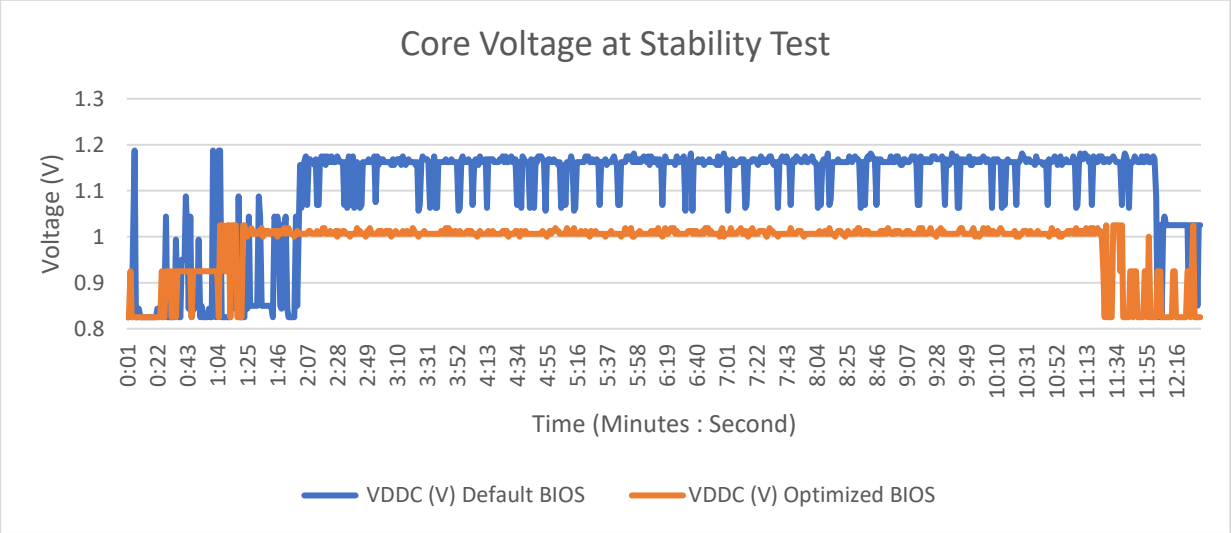

Fig. 5. Core voltage reading at 3DMark stability test

The voltage is capable of meeting the load given with the Modified DVFS three-state voltage shown in Fig. 5. The voltage can adapt to three states, namely at 825 mV, 925 mV, and 1025 mV. Look also at Figure 5 DVFS with three state voltages capable of increasing the stability of the core voltage by lowering the ripple when the GPU is given a test load. The ripples produced are only 25 mV compared to the default BIOS that uses eight DVFS states which produce ripples of 120 mV. DVFS with fewer state voltages can adapt more quickly to systems that run long in idle state or old with high-performance state as done in the stability test. Faster adaptation due to fewer state voltages, so that the transition between idle state or power state medium to high-performance state is faster by passing fewer state stages. Voltage ripples can also be reduced by reducing the voltage state on this DVFS, by minimizing voltage changes due to changes in the computational load

GPU core frequency stability analysis

Frequency ripple occurs in the default condition before modification of the DVFS voltage state when the stability test is performed, but does not occur after the DVFS voltage state is modified. Frequency ripples that occur under default conditions occur up to 40 MHz as shown in graph Figure 6. The core frequency ripples are influenced by the GPU core voltage.

The drop voltage will cause DVFS under default conditions to adjust the core frequency with a readable voltage level. DVFS with eight state voltages has a narrower frequency difference in each state so that the voltage drop will automatically cause a drop in the core frequency. DVFS with three state voltages has a wider voltage range for each frequency state, so that with a slight ripple on the voltage it will not cause a decrease in core frequency.

The core frequency is directly related to GPU computing performance. Decreasing core frequency values will have an impact on the decline of most computing performance. Drops that occur will reduce performance stability and reduce the efficiency of computational performance, therefore increasing core frequency alignment is one step to improve the efficiency of GPU computing performance.

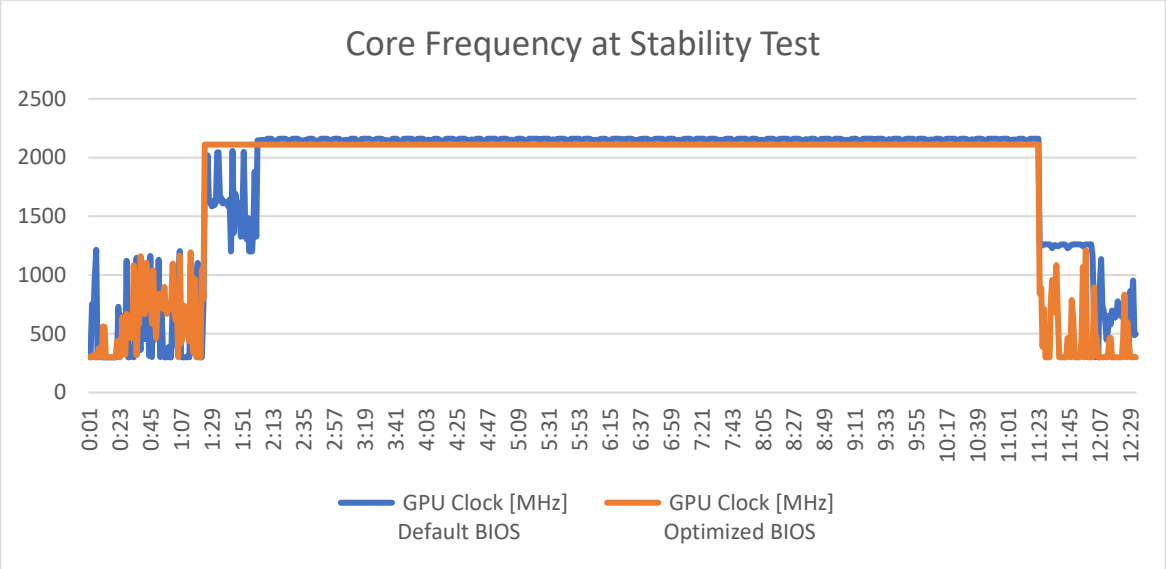

Fig. 6. Core frequency reading at 3DMark stability test

3.4 Computation Test Result and Analysis

Computational science benchmarks in this research use computation, namely: computational algorithm spigot for digit pi calculation through GPUPI software, computational simulation of molecule folding open MM by FAHBenchmark, computation of N-Body simulation by Compubench, execution performance of sobel operator algorithm and histogram equalization algorithm through Geekbench , computational image stitching, SIFT algorithm by Agisoft photoscan, and 3D rendering computation by 3DMark Fire Strike software, and Deep learning using Convolutional Neural Network (CNN).

Table 2. Benchmark results

Benchmark	Result		Memory Allocation (MB)	Difference (%)
	Default BIOS	Optimized BIOS		
Spigot Algorithm	989 seconds	1029 seconds	166	-4.0
OpenMM	63.2255 points	61,5068 points	120	-2.7
GeekBench	118663 points	116162 points	320	-2.1
N-Body Simulation 128k	332.169 iteration/s	340.697 iteration/s	520	2.6
N-Body Simulation 1024k	51.2394 iteration/s	52.5324 iteration/s	650	2.5
3D Render	24166 points	23893 points	1500	-1.1
Image Stitching (669 Gambar)	1211 seconds	1217 seconds	540	- 0.1
Image Stitching (129 Gambar)	141 seconds	141 seconds	620	0
VGG16 (CNN)	16.8 FPS	19.6 FPS	7616	16.7
VGG19 (CNN)	14.9 FPS	16.3 FPS	7571	9.4
MobileNet (CNN)	64.0 FPS	70.6 FPS	4247	10.3

The results of computational performance in Table 2 show a variety of computational performance response responses to optimizing memory timings and decreasing core frequencies. The minus value in Table 2 shows a decrease in computational performance after optimization of

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

memory timings with decreasing frequency values, while a positive value difference indicates an increase in computing performance.

The spigot algorithm is the most effective computing effect on the decrease in core frequency and does not seem to be affected by the optimization of memory timings. N-body simulation is computing that has increased performance even though the core frequency has decreased. Based on the percentage increase, the n-body simulation shows the effect of optimizing the memory timing at the most increase in performance. Computing other than the N-body simulation, decreased performance due to a decrease in core frequency, but the decrease in performance was not as large as the percentage decrease in core frequency. This shows the effect of optimizing memory timings on computing performance.

The tendency of the magnitude of the effect of core or memory on computing can be classified by looking at changes in performance compared to memory usage as shown in Table 2. High memory usage with positive performance changes or indicates an increase in performance with the optimization of memory timings shows the tendency of memory bound. Low memory usage with negative performance changes indicates a decrease in performance where the effect of core frequency reduction is greater than the optimization of memory timings showing core bound tendency. Memory timing optimization will be optimal if the benchmark has a large GPU memory allocation and has low computational intensity so that the effect of data transfer speed on memory will be higher. Data transfer in memory is very dependent on memory frequency and memory timings.

3.5 GPU Output Power and Temperature Analysis

The GPU core is the component that consumes the most power from the entire GPU board. The decrease in core power consumption is due to a decrease in the upper limit of the GPU core frequency on DVFS. Decreasing the upper limit value of voltage in DVFS which is used as a voltage value that runs on high performance state is very influential on GPU power usage.

Table 3. GPU power draw and output temperature

Benchmark	Power Default BIOS	Power Optimized BIOS	Diff (%)	Temperature (Celcius) - Default BIOS	Temperature (Celcius) - Optimized BIOS	Diff (%)
Spigot Algorithm	109.8	76.1	-30.7	73.6	61.4	-16.6
Both GPUs	105.6	77.0	-27.1	64.2	56.2	-12.5
OpenMM	102.2	70.7	-30.8	60.4	52.8	-12.6
Image processing	86.2	58.4	-32.3	43.0	39.6	-7.9
N-Body	116.2	82.1	-29.3	57.0	51.0	-10.5
3D Render (GPU1)	166.8	118.5	-29.0	68.0	58.4	-14.1
3D Render(GPU 2)	161.4	118.7	-26.5	62.6	54.6	-12.8
Image Stitching 669 Images Both GPUs	162.1	112.7	-30.5	88.0	73.0	-17.0
	145.5	110.6	-24.0	79.0	67.0	-15.2
Image Stitching 129 images Both GPUs	156.0	113.1	27.5	79.0	66.0	-16,5
	150.5	103.1	31.5	72.0	61.0	-15.3
VGG16	175.7	122.6	-30.2	58.7	52.0	-11.4
VGG19	174.0	120.4	-30.8	60.3	52.0	-13.8
MobileNet	115.0	80.9	-29.7	49.3	45.7	-7.3

Table 3 shows a decrease in core voltage and a decrease in core frequency on DVFS which can reduce overall GPU power consumption in all computing power outputs.

It can be seen in Table 3 that there is a decrease in the output power modified by the DVFS voltage value compared to the default conditions in all tests which are marked with a percentage that is minus. Core voltage reduction has been proven to reduce the power drawn by the GPU which impacts system power down. GPU power ranges from 24.0% to 32.3% and system power supply ranges from 15.5% to 25.7%. System power is measured in the state of the system in high-performance conditions.

The varying power consumption values in Table 3 show the computational intensity performed at the GPU core. The higher computational intensity will increase the power drawn by the GPU core. The many calculations performed and data transactions will affect power. The core requires higher power when allocation, data size, transaction intensity, and calculation on compute units on the GPU core increases. The variety of power consumption can show the size of the intensity of each computation.

3.6 Performance per Watt

Performance per watt is a measure to see how much the efficiency of a computer component in carrying out its functions. Performance per watt in this study is used as a measure of **the** efficiency of the GPU. The purpose of using performance measures per watt is to see how far the efficiency of BIOS optimization increases. Performance per watt is calculated based on the score of each benchmark divided by the output power. Changes in performance per watt are written as percentages. The power size used is GPU power.

Table 4. Performance per watt

Benchmark	Performance Per Watt		Difference (%)
	Default BIOS	Optimized BIOS	
Spigot Algorithm	150137 digits/s/W	203024 digits/s/W	35.2
OpenMM	0.62 points/W	0.87 points/W	40.3
Image Processing	1377 points/W	1989 points/W	44.4
N-Body Simulation 128K	2.859 iteration/s/W	4.150 iteration/s/W	45.2
N-Body Simulation 1024K	0.441 iteration/s/W	0.640 iteration/s/W	45.1
VGG16 (CNN)	0.096 FPS/W	0.160 FPS/W	66.7
VGG19 (CNN)	0.085 FPS/W	0.135 FPS/W	58.8
MobileNet (CNN)	0.557 FPS/W	0.873 FPS/W	56.7
3D Render	73.6 points/W	100.7 points/W	36.8
Image Stitching 669 images	0.0018 images/s/W	0.0024 images/s/W	33.3
Image Stitching 129 images	0.0030 images/s/W	0.0041 images/s/W	36.7

Increased performance per watt in all tests carried out as shown in table 4. Benchmark computing computing shows an increase in efficiency of 35.2% to 45.2%. Deep learning benchmarks have increased efficiency by 45.8% -66.7%. The lowest increase occurred in the 3D rendering benchmark by 3DMark software, which amounted to 33.3%. The increase in efficiency was due to a combination of power losses caused by core frequency tuning and a decrease in core voltage coupled with GPU performance that was successfully maintained thanks to memory timing optimization.

4. Conclusion

DVFS and memory timing Optimization can increase the stability of GPU core voltage and

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

frequency, decrease the temperature from 7.4% to 17%, decrease power from 19.9% up to 32.6%, and increase the GPU efficiency based on performance per watt from 33.3% up to 66.7% for data center applications.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Y. Arafa, A. -H. A. Badawy, G. Chennupati, N. Santhi and S. Eidenbenz, "PPT-GPU: Scalable GPU Performance Modeling," in IEEE Computer Architecture Letters, vol. 18, no. 1, pp. 55-58, 1 Jan.-June 2019, doi: 10.1109/LCA.2019.2904497

[2] S. Najam, J. Ahmed, S. Masood and C. M. Ahmed, "Run-Time Resource Management Controller for Power Efficiency of GP-GPU Architecture," in IEEE Access, vol. 7, pp. 25493-25505, 2019, doi: 10.1109/ACCESS.2019.2901010.

[3] Y. Ma, J. Zhou, T. Chantem, R. P. Dick, S. Wang and X. S. Hu, "Improving Reliability of Soft Real-Time Embedded Systems on Integrated CPU and GPU Platforms," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 10, pp. 2218-2229, Oct. 2020, doi: 10.1109/TCAD.2019.2940681.

[4] [8] M. Smith, L. Zhao, J. Cordova, X. Jiang and M. Ebrahimi, "Energy-Efficient GPU-Intensive Workload Scheduling for Data Centers," 2023 International Conference on Machine Learning and Applications (ICMLA), Jacksonville, FL, USA, 2023, pp. 1735-1740, doi: 10.1109/ICMLA58977.2023.00263.

[5] S. . -K. Shekofteh, H. Noori, M. Naghibzadeh, H. Fröning and H. S. Yazdi, "cCUDA: Effective Co-Scheduling of Concurrent Kernels on GPUs," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 4, pp. 766-778, 1 April 2020, doi: 10.1109/TPDS.2019.2944602.

[6] C. A. García-Rodríguez, P. Quinto-Diez, J. A. Jiménez-Bernal, L. A. R. -D. León and A. Reyes-León, "Waste Heat Recovery System Applied to a High-Performance Video Card," in IEEE Access, vol. 8, pp. 6272-6281, 2020, doi: 10.1109/ACCESS.2020.2964207.

[7] J. Guerreiro, A. Ilic, N. Roma and P. Tomás, "Modeling and Decoupling the GPU Power Consumption for Cross-Domain DVFS," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 11, pp. 2494-2506, 1 Nov. 2019, doi: 10.1109/TPDS.2019.2917181.

[8] S. M. Nabavinejad, S. Reda and M. Ebrahimi, "Coordinated Batching and DVFS for DNN Inference on GPU Accelerators," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 10, pp. 2496-2508, 1 Oct. 2022, doi: 10.1109/TPDS.2022.3144614.

[9] H. Liu, S. Liu, C. Wen and W. E. Wong, "TBEM: Testing-Based GPU-Memory Consumption Estimation for Deep Learning," in IEEE Access, vol. 10, pp. 39674-39680, 2022, doi: 10.1109/ACCESS.2022.3164510.

[10] C. Zhang, F. Zhang, X. Guo, B. He, X. Zhang and X. Du, "iMLBench: A Machine Learning Benchmark Suite for CPU-GPU Integrated Architectures," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 7, pp. 1740-1752, 1 July 2021, doi: 10.1109/TPDS.2020.3046870.