International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I10P227

Volume 70 Issue 10, 277-284, October 2022
© 2022 Seventh Sense Research Group®

*Original Article*

# Analysing the Energy Value of GPU and Spoting the Energy Hungry Area in the Software Testing Scripts

G. Anithakrishna[1], M. Mohankumar[2]

*[1,2]Department of CS, IT and CA, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India.*

*[1]Corresponding Author : anithamohanvm@gmail.com*

Received: 29 June 2022 Revised: 16 September 2022 Accepted: 12 October 2022 Published: 20 October 2022

***Abstract*** *- Nowadays, the Graphic Processing Unit, GPU become a pronounced tool for individual and business computing. GPU plays a great role in a wide range of areas like parallel processing, video rendering, graphics, gaming and artificial intelligence. This work analyses the performance and energy efficiency of designing manual and automated scripts for game development and prominent video and graphics applications. These days' software games have prolonged lifetimes and have many patches and releases compared to the olden days. Nowadays green concept has a very important role in developing efficient software. Software development is cognate with different phases, which include a broad range of activities. Software metrics are some techniques that enable the analysis of code and its improvement. A powerful Graphic Processing Unit is compulsory for executing upscale games and applications that use 3D and video editing. This paper aspires to monitor GPU's performance and power concern for video rendering and game development and to spot the energy-hoggish area in the script using the thread concept.*

***Keywords*** *- Green software, GPU, Software Testing, Sustainability, Green IT, Energy efficiency.*

## 1. Introduction

Generally, people consider GPU for gaming and scenarios like video rendering. These applications require computations to quickly represent curves, polygons and different shapes on the screen, which is difficult in real-time using a CPU. It is a matter of embarrassingly parallel means of computation to handle large batches of vertices and large blocks of pixels to render image processing like raytracing of 3D images etc.

The working of the GPU compared to the CPU is almost the same except for processing graphics-related data and functions. In integrated graphics deploying both CPU and GPU will limit one of their prospective processing powers, whereas a dedicated graphics card has its circuit board, processor, memory and cooling systems, which will reduce the workload of the main processor. Most applications carted on GPU are parallel with data independence and have no synchronisation on execution threads.

CPU and GPU, the brain and soul are the key factors in modern computing and breaking a complex problem into several millions of tasks and handling it through [26] parallel computing makes GPU more powerful. Figure 1 depicts the difference between the Central Processing Unit and Graphics Processing Unit, and It's clear that the CPU remains essential, fast and versatile.

The power consumption of GPU has a remarkable impact on solidity, economic attainability, performance and arrangements in a broad range of applications. Certain management techniques are vital for the CPU and GPU to handle the power dissolution. Power management on GPU is an effective step towards green.

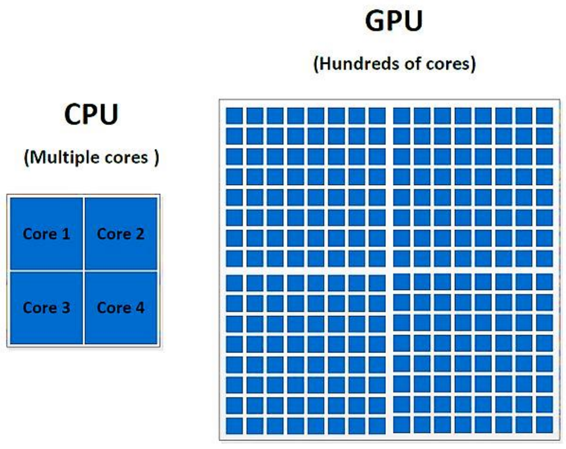


**Fig. 1 CPU and GPU**

This work aims to analyse GPU's energy efficiency and spot a script's energy-hungry area. A GPU has several streaming multiprocessors with multiple cores [10]. Now the latest NVIDIA GeForce RTX 30 series graphics cards have the fastest ray tracking cores and tensor cores, providing eminent quality for images and videos and excellent frame speed.

## 2. Literature Review

Several tools and models have recently been proposed to measure GPU power consumption. Caroline Collange and David Defour investigated Nvidia GPU for general purpose in a CUDA environment [3], identified the consumption and characterised it using physical measurement.

Hong and Kim proposed a power and performance modelling system for general-purpose GPU and focused on active processors [7] for an application. Luo and Suda proposed an execution time and energy consumption model for [14] parallel mass programs.

Rofouei and Ryffel conducted a study on energy cost in GPU operations [17] compared to CPU solely systems. Hamada and Benkrid performed a comparative study on general purpose processors,[27] GPU and FPGA based on some criteria. It includes performance, speed, power and cost development. Liamocca and Carranza compared FPGA and GPU for video processing programs and found the efficiency of GPU over FPGA due to its mass parallel processing feature.

Lange compares the multi-core CPU, GPU and FPGA performance for geometric algebra calculations using a reconfigurable [20] computer. Chandramowlishwaran and Williams analysed and compared the efficiency [4] of GPU with multi-core CPU and analysed the fast multipole method. Mcintosh and Smith focused on real-time data to compare [15] multi-core CPU and GPU for molecular mechanism problems.

## 3. Green energy-efficient GPU

Energy management is important for the performance and reliability of the computing system. Power constraints are important for the simultaneous usage of cores in a mass parallel processing system. Sustainable computing, which causes less carbon emission from ICT devices, is a major concern, and researchers place giant prominence on energy efficiency in the design of computing systems [21, 25]. Energy-efficient components are important for achieving environmentally sustainable products.

## 4. Related works

In the early stage, GPUs have limited functions like 3D graphics rendering and subsequently, 3D graphic APIs like Open Graphics Library wide open new features to the developers. As time passes, programmers need to create more complex and special visual effects on the screen, and GPU's development domain has extended enormously. This section will discuss some early-stage studies to improve the energy efficiency of GPU.

### *4.1. Power Usage and Control Mechanisms*

Power is a time-related quantity; for GPU, total power is the sum of idle power, multiprocessor power, and memory power. It can be summarised as

$$\text{Total Power} = \text{Multiprocessors Power} + \text{Memory Power} + \text{Idle Power} \qquad (1)$$

Here Multiprocessor power is the sum of power used by each streaming multiprocessor.

### *4.2. DVFS based approach*

Earlier, Dynamic Voltage and Frequency Scaling are used to reduce energy consumption. The power and frequency are related as

$$P \propto F\,V^2 \qquad (2)$$

Here F stands for frequency, and V stands for voltage. Bringing down the frequency leads to saving power, but it should ensure that it will not affect the performance. Nam and Yoo proposed a GPU which uses less power for logarithmic calculations. They applied DVFS to the processor and rendering engine and supplied frequency drawn on workload. Anzt and Heuveline studied an iterative linear solution using DVFS, which provides large energy savings with small performance [12] dropping.

### *4.2. Energy Efficiency in GPU elements*

Various techniques are applied in the structure to save component-level energy usage, which utilises components' run time transformations. Lashgar and Benjasadi exploited Instruction [16] Temporal Locality in GPU to explore the filter cache instead of the instruction cache.

Gebhart and Johnson proposed a mechanism to reduce the energy usage of GPU for on-chip memory for the register storage [1] of threads. Here they implemented a tiny storage structure to the register files and provided prioritisation to the active threads by the scheduler, which pilots notable energy savings.

Lee and Kim proposed thread-level parallelism since [8] GPU has more threads than CPU. Hence applied, a core sampling technique to evaluate cache on GPU performance.

### *4.3. Code Level Techniques*

Through many research works, it has been observed that code-level optimisation has a significant role in energy savings and CPU and GPU performance [22]. So, optimising the GPU implementation can lead to great energy savings. Ghosh and Chandrasekharan studied high performance computing on GPU and multi-core CPU. They recognised that GPU has a significant role in power consumption for global memory access. Yang and Mandor analysed many GPU-based applications [6], selected works from various disciplines, and traced certain patterns that will degrade the hardware performance. Optimising the source code for GPU-specific features will provide energy efficiency.

## 5. Automated Testing and Validation for Game Development

Nowadays, trend-setting video games have become more creative and interdisciplinary and need to board new technologies regularly. Here the targeted clients are from multiple platforms, and we should ensure that it performs as expected in each platform. So, the testing activity is very important and is performed on various hardware setups, including a wide class of CPU, GPU, RAM and drivers. A game released with even a minor defect negatively impacts goodwill and commercial bang [23]. Murphy-Hill and Zimmermann conducted a study to compare video game development [5] and other software development and motivated researchers to create testing tools for game developers. Cooper and Scacchi focused on various software testing areas in video game [9] development which enabled automated testing.

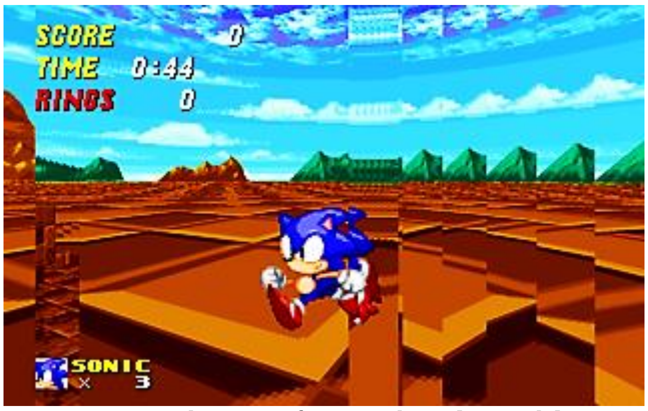


**Fig. 2 Visual error in software rendering [SRB2wiki]**

Flaws in an in-game application can emerge from conventional code-level programming mistakes, amendments in the game content or difference in the hardware exemplar of developer and consumer. Uncovering and rectifying defects is a time-consuming and expensive activity in the software development process [24], and finding bugs in Graphical User Interface is comparatively difficult. Mixed bags of commercially released games reported graphical defects through the game company forum and user's community, which led to post-release updates.

Quality analysts categorised bugs in the video game based on severity and prioritised them into various categories. Based on priority, bugs are classified into various classes like A, B, and C. Correct categorisation and prioritisation of bugs can eliminate extra effort in locating and rectifying the defects. Lewis and Whitehead presented a taxonomy that categorises [11] invalid representation of graphical data, lack of information, invalid data access and response issues as different bugs and has different subcategories. The hall of mirror effect is an example of a visual defect in software rendering due to missing texture or outside boundary. Figure 2 depicts the effect of the hall of mirror defect. Automated testing will reduce testing time and testers' overhead [18] compared to manual testing.

## 6. Automated and Manual Video Testing

A web application with a lot of video content needs a visual video testing technique to check the correctness. To check the quality of the video to play, select the time to capture a frame in the video, the video's play length, and correctness while converting the video to another format like WAV, MOV, MP4 etc. Visual aberrations are tiring to detect, so to ensure visually flawless applications, we need an automated framework to detect and correct them. Management of UI components and their perfect rendering on various platforms and devices is highly prioritised because customer experience depends on what they see and makes a difference.

## 7. Testing on VR Applications

Virtual Reality enables sound drenching in simulated domains through which customers can interact. Time frame constraints to use VR devices, late application access to the test phase in the development life cycle, and multi-level analysis on multiple platforms are the major obstacles in the testing activity.

## 8. Approach

This section describes the overview and scope of the work to evaluate and analyse the energy efficiency of GPU, memory, and CPU over various test suits. Table 1 depicts the system configuration.

For implementing green metrics in the software testing process focuses on the software part, which causes hardware components to consume more energy which leads negative impact on the environment as the first step is to select the test script and analyse the performance and energy usage for the execution of the selected application.

**Table 1. Platform configuration**

| | |
|---|---|
| **CPU** | Intel(R) Core (TM) i7-7820HQ |
| **Clock rate** | 2.90GHz |
| **RAM** | 16.0GB |
| **Main Board** | Dell Inc. -0R6JFH, A00 |
| **Hard Disk** | MTFDDAV512TBN-1AR1ZABH |
| **GPU** | NVIDIA Quado M1200 |
| **Memory clock rate** | 2901 |

## 9. Experimental Procedure

Software testing aims to ensure the software is bug-free and to release quality products to the client. It ensures that the system meets customer requirements, specifications and end-user expectations. To experiment, two demo projects are selected, "fountoTuto", a web application for innovative personalised private tutoring and "travoaide", a travel guidance platform for planning, reserving and proceeding to a trip. Figure 4 shows test cases generated for the manual test of "fountoTuto", a web application for innovative personalised private tutoring. We have conducted an experiment to analyse energy consumption and performance of GPU, CPU and memory resources on the software test phase and implemented some manual test cases and automated test scripts using the selenium testing tool.

Fig. 5 shows the Selenium test script for "travoaide". Selenium is a powerful testing tool suite of tools such as an Integrated Development Environment, Remote control, Web driver and Selenium Grid.

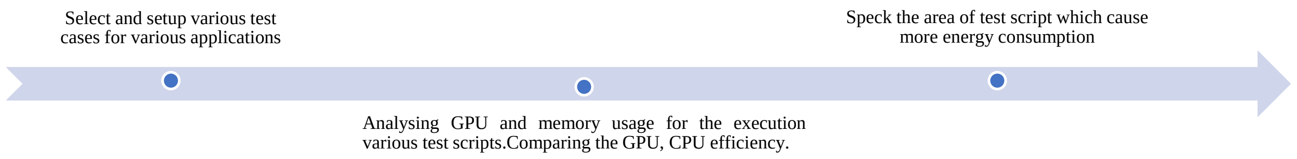

Select and setup various test cases for various applications
Speck the area of test script which cause more energy consumption
Analysing GPU and memory usage for the execution various test scripts.Comparing the GPU, CPU efficiency.

**Fig. 3 Steps towards green testing activity**

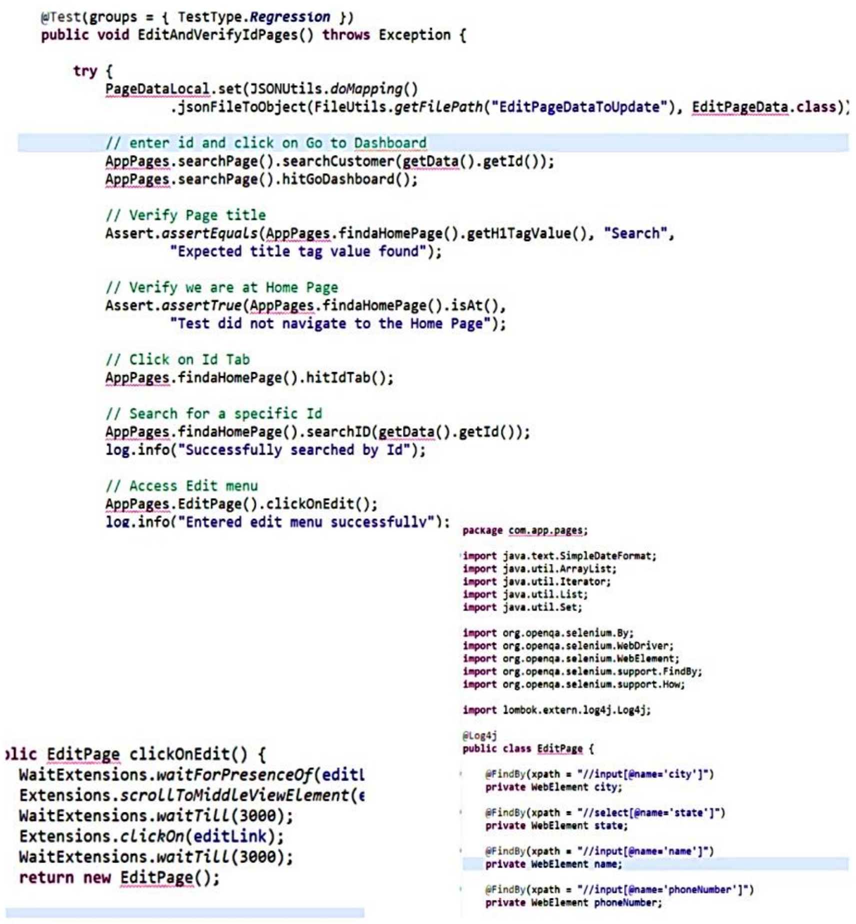
| Subject | Test Case Name | Test Case Description | Step # | Step Description | Expected Result | Input Parameters | Designer | Status | Priority | Type | Reviewer * |
|---|---|---|---|---|---|---|---|---|---|---|---|
| founto Tuto before sign in | TC_001 | Verify that User can able to see the videos in home page or not | 1 | Give url for the founto Tuto in the browser | The founto Tuto should be loaded | | Anitha | | High | Manual | <<reviewer name>> |
| | | | 2 | Check Whether the videos are displayed or not | Videos should display | | | | | | |
| founto Tuto before sign in | TC_002 | **Verify that the user is able to Scroll videos or not.** | 1 | Give url for the founto Tuto in the browser | The founto Tuto with video should be loaded | | Anitha | | Medium | Manual | <<reviwer name>> |
| | | | 2 | Click on the Scrollbar and move it | Videos should scroll according to the movement of the scrollbar | | | | | | |
| founto Tuto before sign in | TC_003 | **Verify that the user can pause the video or not** | 1 | Give url for the founto Tuto in the browser | The founto Tuto with video should be loaded | | Anitha | | Medium | Manual | <<reviewer name>> |
| | | | 2 | Click on the video or pause | Videos should pause | | | | | | |
| founto Tuto before sign in | TC_004 | **Verify that the user can mute the video or not.** | 1 | Give url for the founto Tuto in the browser | The founto Tuto with video should be loaded | | Anitha | | Medium | Manual | <<reviewer name>> |
| | | | 2 | Click on the mute button | Videos should mute | | | | | | |
| founto Tuto before sign in | TC_005 | **Verify that the user is able to increase and decrease the volume or not.** | 1 | Give url for the founto Tuto in the browser | The founto Tuto with video should be loaded | | Anitha | | Medium | Manual | <<reviewer name>> |
| | | | 2 | Change the volume of the video | Videos should be change accordingly | | | | | | |
| founto Tuto before sign in | TC_006 | **Verify that the user is able to click on a mini player** | 1 | Give url for the founto Tuto in the browser | The founto Tuto with video should be loaded | | Anitha | | Medium | Manual | <<reviewer name>> |
| | | | 2 | Click on the mini player | After clicking on it, the screen should be minimized. | | | | | | |
| founto Tuto before sign in | TC_007 | **Verify that the user is able to select the video quality from the list.** | 1 | Give url for the founto Tuto in the browser | The founto Tuto with video should be loaded | | Anitha | | Medium | Manual | <<reviewer name>> |
| | | | 2 | Select video quality from the list. | User should be able to select the quality and video should be displayed with the selected quality | | | | | | |

**Fig. 4 Manual Test Case designed for the "fountoTuto" application**

Our proposed tool is Linux based, which estimates GPU, CPU and Memory utilisation of the testing activity. It continuously observes the energy usage and saves the output in a comma-separated value format. Test scripts are generated for the least graphic application and a video-enriched application to analyse the performance. Continuously monitor and save the resource usage into a file by the process script, which will iterate and filter out the specific process and appends the result to the file. While performing the testing process, we ran the above shell script and stored the data usage as a csv file. To examine code's role in energy consumption, scrutinised the per thread CPU usage. The thread concept helps us slice, detail, and find the energy-hungry area of the process.

## 10. Result

This section describes the evaluation of the result generated during the experiment. The result gathered in the csv file is organised, and the graph is plotted. Graphs show the usage of GPU, CPU and memory during the automated test process of the "travoaide" application.



```
@Test(groups = { TestType.Regression })
public void EditAndVerifyIdPages() throws Exception {

    try {
        PageDataLocal.set(JSONUtils.doMapping()
                .jsonFileToObject(FileUtils.getFilePath("EditPageDataToUpdate"), EditPageData.class));

        // enter id and click on Go to Dashboard
        AppPages.searchPage().searchCustomer(getData().getId());
        AppPages.searchPage().hitGoDashboard();

        // Verify Page title
        Assert.assertEquals(AppPages.findaHomePage().getH1TagValue(), "Search",
                "Expected title tag value found");

        // Verify we are at Home Page
        Assert.assertTrue(AppPages.findaHomePage().isAt(),
                "Test did not navigate to the Home Page");

        // Click on Id Tab
        AppPages.findaHomePage().hitIdTab();

        // Search for a specific Id
        AppPages.findaHomePage().searchID(getData().getId());
        log.info("Successfully searched by Id");

        // Access Edit menu
        AppPages.EditPage().clickOnEdit();
        log.info("Entered edit menu successfully");
```

```
package com.app.pages;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;

import lombok.extern.log4j.Log4j;

@Log4j
public class EditPage {

    @FindBy(xpath = "//input[@name='city']")
    private WebElement city;

    @FindBy(xpath = "//select[@name='state']")
    private WebElement state;

    @FindBy(xpath = "//input[@name='name']")
    private WebElement name;

    @FindBy(xpath = "//input[@name='phoneNumber']")
    private WebElement phoneNumber;
```

```
lic EditPage clickOnEdit() {
  WaitExtensions.waitForPresenceOf(editL
  Extensions.scrollToMiddleViewElement(e
  WaitExtensions.waitTill(3000);
  Extensions.clickOn(editLink);
  WaitExtensions.waitTill(3000);
  return new EditPage();
```

**Fig. 5 Test Script written for the "travoaide" project**

```
Algorithm
Step 1:  Delete the output .csv file <<OutputFile.csv>> if exists.
Step 2:  Create the output .csv file <<OutputFile.csv>>
Step 3:  Print Caption "Output of CPU, Memory, GPU Usage"
Step 4:  Print Heading "TIME_STAMP, CPU_USAGE%, MEMORY_USAGE%, GPU_USAGE%"
Step 5:  Start Iteration
Step 6:  Format Date with Time Stamp
Step 7:  Filter the CPU usage for given program that is being executed
Step 8:  Calculate the Memory usage by considering the total memory utilisation and the memory usage
for given program that is being executed
Step 9:  Filter the GPU usage for given program that is being executed
Step 10:  Append the file with the data from the steps 6, 7, 8 and 9 separated with comma.
Step 11:  Pause iteration the for 1 second
Step 12:  Repeat from Step 6:
Step 13:  Stop test process
Step 14:  Locate the folder for <<OutputFile.csv>> file
```

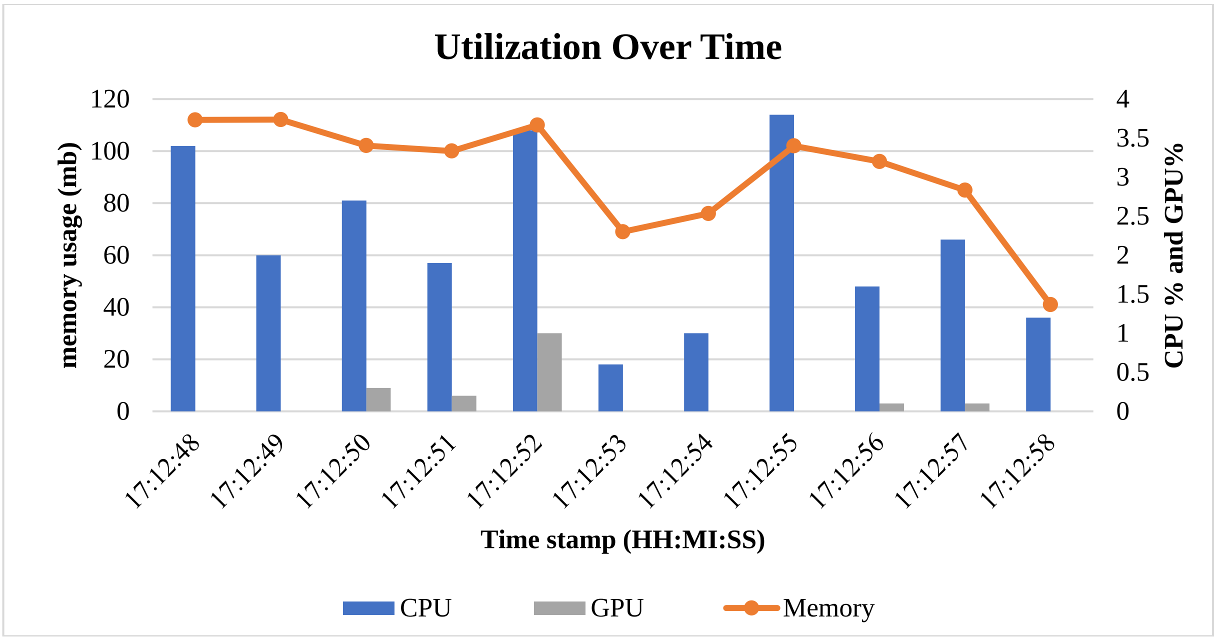


Fig. 6 Graphical representation of CPU, GPU and MEMORY usage on testing the "travoaide" project

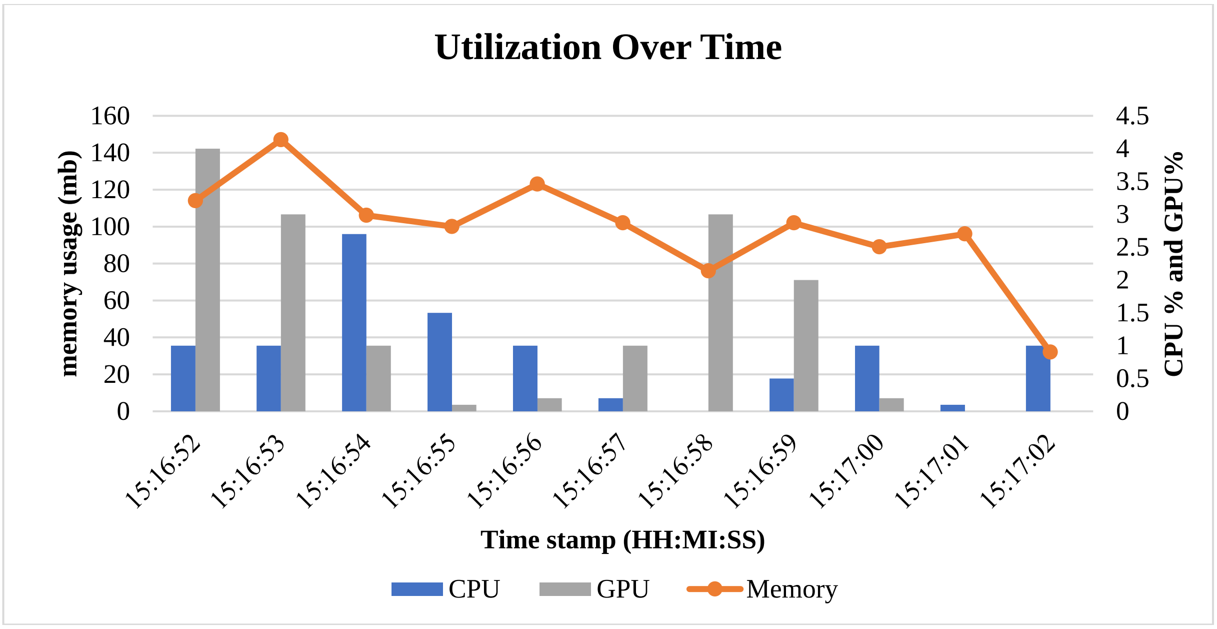


Fig. 7 Resource Usage on Testing the "fountoTuto" application

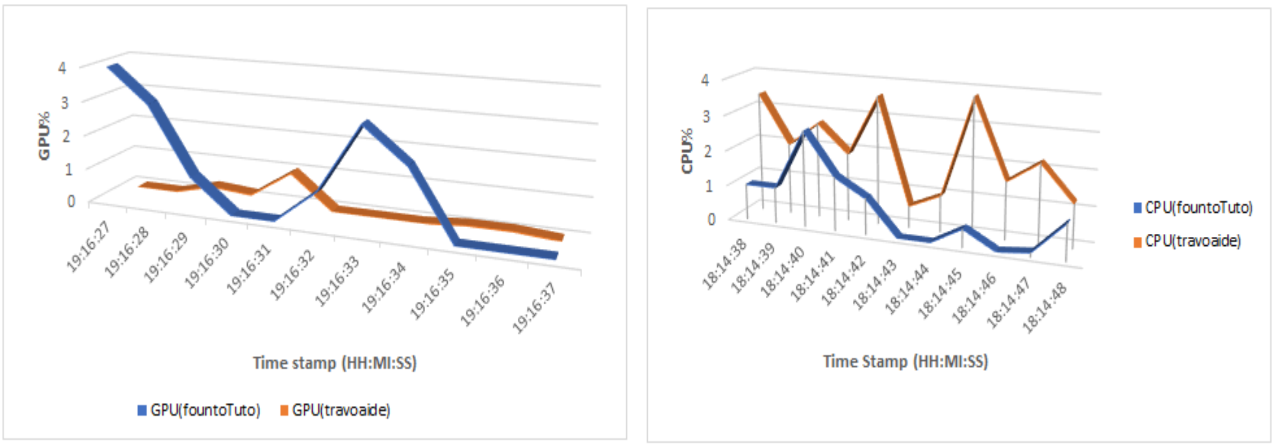


Fig. 8 Comparison of GPU and CPU usage for both "fountoTuto" and "travoaide"

For evaluation, considered two applications, one with the least graphical interfaces and one supplemented with videos. The Linux-based script analysed the performance while testing the application. The continuous usage of these resources has a major influence on energy consumption.

Fig. 6 illustrates the resources' usage while testing "travoaide" The graph is plotted with the timestamp in HH:MI: SS on the x-axis and memory in megabytes on the y-axis. The usage percentage of CPU and GPU readings are indicated on the right axis of the graph. Here GPU usage is less compared to CPU. Figure 7 exemplifies the memory usage on the y-axis and CPU and GPU usage on the right axis of the graph. Each one is plotted against the x-axis, which indicates the timestamp for the "fountoTuto" project. This application includes more video and graphical content; thus, GPU usage increased here compared to previous test scenarios. Graphs are plotted by analysing the csv file mapped the result while testing.

GPU has primacy over CPU in energy efficiency in parallel processes and those applications with concerted computations. Assigning all workloads to GPU is not an efficient solution, and that will lead to CPU idling. Proper issuance of resources and optimised utilisation of same can procure green sustainable development. In Figure 8, the First graph shows the GPU usage of two applications, "fountoTuto" and "travoaide" x-axis is plotted with the timestamp in HH:MI: SS and the y-axis is plotted with a percentage of GPU usage, whereas the second graph illustrates the comparison of CPU usage of the two applications.

## 11. Conclusion and Future Work

As technologies grow, the use of related devices greatly impacts the environment. A single individual or a big organisation both have some responsibility towards the environment to promote green principles. Green computing is the practise of using computing resources efficiently by reducing the technology's negative effect on the environment.

The sight of green computing is to reduce carbon emissions by reducing resource usage and energy efficiency of CPU, GPU and other peripherals. The future of this chore is to unroll the scope of code complexity analysis, which will lead to an environment- friendly software development Life cycle. The system with many complex interfaces and requirements is extravagant to debug and maintain. Evaluating the code complexity is an appropriate tool to foresee the defect possibility.

## References

[1] Gebhart, Mark, Daniel R. Johnson, David Tarjan, Stephen W. Keckler, William J. Dally, Erik Lindholm, and Kevin Skadron, “Energy-Efficient Mechanisms for Managing Thread Context In Throughput Processors,” In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 235-246. IEEE, 2011.

[2] Lee, Jeabin, Byeong-Gyu Nam, and Hoi-Jun Yoo, “Dynamic Voltage and Frequency Scaling (DVFS) Scheme for Multi-Domains Power Management,” In *2007 IEEE Asian Solid-State Circuits Conference*, IEEE, pp. 360-363, 2007.

[3] Collange, Sylvain, David Defour, and Arnaud Tisserand, “Power Consumption of Gpus From A Software Perspective,” In *International Conference on Computational Science*, Springer, Berlin, Heidelberg, pp. 914-923, 2009.

[4] Chandramowlishwaran, Aparna, Samuel Williams, Leonid Oliker, Ilya Lashuk, George Biros, and Richard Vuduc, “Optimising and Tuning the Fast Multipole Method for State-of-The-Art Multi-Core Architectures,” In *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, . IEEE, pp. 1-122010.

[5] Murphy-Hill, Emerson, Thomas Zimmermann, and Nachiappan Nagappan, “Cowboys, Ankle Sprains, and Keepers of Quality: How is Video Game Development Different from Software Development?,” In *Proceedings of the 36th International Conference on Software Engineering*, pp. 1-11. 2014.

[6] Yang, Yi, Ping Xiang, Mike Mantor, and Huiyang Zhou, “Fixing Performance Bugs: An Empirical Study of Open-Source GPGPU Programs,” In *2012 41st International Conference on Parallel Processing*, pp. 329-339. IEEE, 2012.

[7] Hong, Sunpyo, and Hyesoon Kim, “An Integrated GPU Power and Performance Model,” In *Proceedings of The 37th Annual International Symposium on Computer Architecture*, pp. 280-289. 2010.

[8] Lee, Jaekyu, and Hyesoon Kim, “TAP: A TLP-Aware Cache Management Policy for A CPU-GPU Heterogeneous Architecture,” In *IEEE International Symposium on High-Performance Comp Architecture, IEEE*, pp. 1-12, 2012.

[9] Scacchi, Walt, and Kendra M. Cooper, “Research Challenges At The Intersection of Computer Games and Software Engineering,” *In Proceedingd 2015 Conference Foundations of Digital Games*, 2015.

[10] Ghosh, Sayan, Sunita Chandrasekaran, and Barbara Chapman, “Energy Analysis of Parallel Scientific Kernels on Multiple Gpus,” *In 2012 Symposium on Application Accelerators in High-Performance Computing*, pp. 54-63. IEEE, 2012.

[11] Lewis, Chris, Jim Whitehead, and Noah Wardrip-Fruin, “What Went Wrong: A Taxonomy of Video Game Bugs,” *In Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pp. 108-115. 2010.

[12] Anzt, Hartwig, Vincent Heuveline, José I. Aliaga, Maribel Castillo, Juan C. Fernandez, Rafael Mayo, and Enrique S. Quintana-Orti, “Analysis and Optimisation of Power Consumption in the Iterative Solution of Sparse Linear Systems on Multi-Core and Many-Core Platforms,” In *2011 International Green Computing Conference and Workshops*, pp. 1-6, IEEE, 2011.

[13] Hanan Qassim Jaleel, "Testing Web Applications," *SSRG International Journal of Computer Science and Engineering*, vol. 6, no. 12, pp. 1-9, 2019. *Crossref*, https://doi.org/10.14445/23488387/IJCSE-V6I12P101.

[14] Kim, Gloria YK, Akihiro Hayashi, and Vivek Sarkar, “Exploration of Supervised Machine Learning Techniques for Runtime Selection of CPU Vs GPU Execution In Java Programs,” *In International Workshop on Accelerator Programming Using Directives*, Springer, Cham, pp. 125-144, 2017.

[15] Mcintosh-Smith, Simon, Terry Wilson, Amaurys Ívila Ibarra, Jonathan Crisp, and Richard B. Sessions, “Benchmarking Energy Efficiency, Power Costs and Carbon Emissions on Heterogeneous Systems,” *The Computer Journal,* vol. 55, no. 2 , pp. 192-205, 2012.

[16] Lashgar, Ahmad, Amirali Baniasadi, and Ahmad Khonsari, “Inter-Warp Instruction Temporal Locality In Deep-Multithreaded Gpus,” In *International Conference on Architecture of Computing Systems*, Springer, Berlin, Heidelberg, pp. 134-146, 2013.

[17] Rofouei, Mahsan, Thanos Stathopoulos, Sebi Ryffel, William Kaiser, and Majid Sarrafzadeh, “Energy-Aware High Performance Computing With Graphic Processing Units,” *In Workshop on Power Aware Computing and System*, 2008.

[18] Petrillo, Fábio, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich, “Houston, We Have A Problem..A Survey of Actual Problems in Computer Games Development,” In *Proceedings of The 2008 ACM Symposium on Applied Computing*, pp. 707-711. 2008.

[19] Ramya D, Ramyashree P R, Sunaina Rashmi R, Nalina V , "*Green Cloud Computing,”* A Review - *International Journal of Recent Engineering Science,* vol. 5, no. 6, pp. 16-18, 2018. Http://Ijresonline.Com/Archives/IJRES-V5I6P103.

[20] Lange, Holger, Florian Stock, Andreas Koch, and Dietmar Hildenbrand, “Acceleration and Energy Efficiency of A Geometric Algebra Computation Using Reconfigurable Computers and Gpus,” In *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines, IEEE,* pp. 255-258, 2009.

[21] Muthu, Mohankumar, K. Banuroopa, and S. Arunadevi, “Green and Sustainability In Software Development Lifecycle Process,” *Sustainability Assessment at the 21st Century,* vol. 27, no. 63 , 2019.

[22] Anithakrishna, G., Mohankumar, M, “SEFGAST: Step-Up to Environment Friendly Green Automated Software Testing,” *International Journal of Engineering Trends and Technology,* vol. 70, no. 3, pp. 162-169, 2022.
Crossref, https://doi.org/10.14445/22315381/IJETT-V70I3P218.

[23] Rosziati Ibrahim, Ammar Aminuddin Bani Amin, Sapiee Jamel, Jahari Abdul Wahab, "Epit: A Software Testing Tool for Generation of Test Cases Automatically," *International Journal of Engineering Trends and Technology* vol. 68, no. 7, pp. 8-12, 2020. Crossref, https://doi.org/10.14445/22315381/IJETT-V68I7P202S.

[24] Bijendra Singh, Dr. Ankit Kumar, Dheeraj Kumar Sahni, Divya Shree, Anu, Khushboo, Kapil Sirohi, Dhiraj Khurana ,"A Model to Measure Software Testing Effort Estimation in the Integrated Environment of ERNN, BMO & PSO,*" International Journal of Engineering Trends and Technology,* vol. 69, no. 8, pp. 81-88, 2021. Crossref, https://doi.org/10.14445/22315381/IJETT-V69I8P210.

[25] Sandhya, Nidhi B.Satija, Priyank Singhal, “Green and Sustainable FPGA Based Counter for IOT Based Processor,” *International Journal of Engineering Trends and Technology*, vol. 67, no. 9, pp. 51-54, 2019.
Crossref, https://doi.org/10.14445/22315381/IJETT-V67I9P208.

[26] Llamocca, Daniel, Cesar Carranza, and Marios Pattichis, “Separable FIR Filtering in FPGA and GPU Implementations: Energy, Performance, and Accuracy Considerations,” In *2011 21st International Conference on Field Programmable Logic and Applications*, *IEEE,* pp. 363-368, 2011.

[27] Hamada, Tsuyoshi, Khaled Benkrid, Keigo Nitadori, and Makoto Taiji, “A Comparative Study on ASIC, Fpgas, Gpus and General Purpose Processors in the O (N^ 2) Gravitational N-Body Simulation,” *In 2009 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE,* pp. 447-452, 2009.