Benchmark review and case study of stress test for a gaming computer applied in CPU

Tamires Martins Rezende
FITec
Belo Horizonte - MG, Brasil
tamiresrezende@fitec.org.br

Alfredo Enrique Macias Medri
FITec
Manaus - AM, Brasil
alfredomedri@fitec.org.br

Jacqueline Teixeira Santos
FITec
São José dos Campos - SP, Brasil
jacquelinetsantos@fitec.org.br

João Ider Silva Junior
FITec
São José dos Campos - SP, Brasil
joaoiderjunior@fitec.org.br

Marcos Antonio Alves
FITec
Belo Horizonte - BH, Brasil
marcosaalves@fitec.org.br

Fabio de Carvalho Zottino
FITec
São José dos Campos - SP, Brasil
fczottino@fitec.org.br

ABSTRACT

A stress test on gaming computers is one of the instruments used to ensure that machines can handle demanding memory, processing, and graphics tasks without compromising performance. This study examines benchmark tools for stress testing both personal and gaming computers, as well as degradation variables that can impact their performance. A C++ algorithm based on prime number calculation was created to stress and measure the consumption of three CPUs: AMD Ryzen 7 4800H (Nitro 5 AN515-44), 12th Gen Intel(R) Core(TM) i7-12700H (Predator Triton 300 SE), and 12th Gen Intel(R) Core(TM) i9-12900H (Predator Triton 500 SE). To provide similar conditions, the test was run with ten simultaneous threads on each machine. The results show good performance of the algorithm and revealed the superior performance of the Predator Triton 500 SE in terms of runtime. The discussion addresses the findings that can be utilized to identify potential bottlenecks and ensure a positive gaming experience.

CCS CONCEPTS

• **General and reference** → **Performance**; • **Hardware** → **Hardware test**.

KEYWORDS

benchmark test, stress test, gaming computer, CPU

1 INTRODUCTION

A gaming computer is a high-performance computing device designed for memory-intensive, processing-heavy, and graphics-rich activities. The positive experiences of players drive the continuous growth in the gaming market and works on organizational strategy in the digital games industry [16].

Performance evaluation tests are conducted on these machines to assess system stability and performance under extreme loads. These tests create scenarios in which the computer is pushed to its limits, such as simulations running intensive games, ensuring that the system operates efficiently and reliably in challenging situations [22]. These performance comparison programs, known as benchmarks, help identify speed and performance bottlenecks. The tools may be used, for instance, to extract information for comparison with similar setups. The main performance metrics in the industry of digital games generally include Central Processing Unit (CPU) usage, temperature, clock speed, voltage, and system stability; Graphics Processing Unit (GPU) usage and performance in games; Random-Access Memory (RAM) usage and speeds; and Hard Drive (HD) speeds and access times.

Benchmarks have been widely utilized in academic research to evaluate computing performance. Eigenmann [7] conducted research on methodology, the interaction between SPEC benchmarks and other operations, shortcomings, and the necessity for more initiatives. The authors pointed out that understanding and enhancing computer system performance evaluation requires a focus on these areas. This is in line with [21] study, whose research evaluated the comparison between two processors (Intel and AMD). These authors argued that this type of research is important given the demand for greater computational power in more recent applications.

CPU performance, specially in gaming computers, is an important component in delivering an enjoyable player experience. As gaming technology evolves, the demand for high-performance CPUs has intensified, making it essential for gamers to understand their hardware's capabilities. The literature reports several tools for conducting stress tests that analyze different components and providing machine characteristics, along with a score indicating how good the computer performs for gaming is. However, these tools are often opaque, and their scoring system is difficult to understand. The score is a kind of a metric that represents an overall assessment of the computer or component's performance. Unfortunately, in most cases, this score is calculated using a black-box approach, with varying score ranges across different tools. While it is acknowledged that many variables can contribute to an overall score, these tools typically do not clearly specify how the scores are derived.

In this paper, we conducted a benchmark review and case study of CPU stress testing for gaming computers. Our aim is to present and discuss various benchmark tools for stress tests, highlight degradation factors that can impact performance of computers under extreme loads, and provide a C++ algorithm based on prime numbers, which we apply to measure the CPU performance of three different gaming machines.

This work fits into this event for two main reasons: First, the analysis of benchmark tools provides a straightforward approach that can be utilized by the gaming industry, players, and the event audience. Second, the score provided serves as a metric for evaluating the machine's performance in various aspects, directly aligning

with the topic of "performance, load, and stress tests." The proposed solution in this paper directly addresses these challenges by presenting a C++ algorithm designed to stress test CPUs effectively. Unlike existing tools that operate in a black-box manner, our approach aims to provide clear and interpretable performance metrics based on a systematic analysis of CPU behavior under load.

The rest of this paper is organized into the following sections: Section 2 discusses related works. Section 3 details the methodology utilized to evaluate the different machines. Section 4 presents and discuss the results. Finally 5 highlights the conclusion and future works.

2 RELATED WORKS

The literature presents several studies in stress tests for gaming computers. Sibai [17, 18] presented an analysis and evaluation of personal computer performance in terms of CPU, memory, and graphics using the OSMark benchmark. The study focused on specific tests and compares the benchmark results of two different personal computers with single and dual cores. The author argued that although the tool provides relevant information about the tests, it falls short in total instruction and application coverage, and does not adequately stress multi-core processors compared to the competing benchmarks. Similarly, Ivanova et al. [10] utilized various benchmark tools to evaluate GPUs by incorporating synthetic benchmarks such as 3DMark, 3DMark11, and Unigine Heaven. The authors reported interesting results and pointed out the importance of development next-generation architectures.

In addition to the importance of component analysis for players, Ivanova et al. [10] explored the energy-intensive nature of digital games and offers insights into reducing energy consumption. Significant variations in the energy ratings of computer components suggest a considerable potential for energy savings.

CPU stress testing software, such as Prime95 and Cinebench R23, thoroughly evaluates CPU performance and stability under extreme loads. These tools offer detailed insights into CPU utilization, temperatures, and other critical metrics. However, because they exclusively focus on CPU testing, they may not provide a complete assessment of the system. This could result in outcomes that do not fully represent real-world gaming performance [13, 19, 23].

Similarly, GPU stress testing software such as 3DMark, Heaven / Valley Benchmarks, and GPU-Z simulate intense graphics workloads to stress GPU. These tools provide detailed information about GPU performance, temperatures, and stability. However, because they primarily focus on testing the GPU, they may not offer a complete picture of the overall system performance. Results from these tests do not always correlate with actual gaming performance due to the unique characteristics of gaming workloads [13, 19, 23]. In terms of GPU application performance, Che and Skadron [4] emphasize the relevance of benchmark correlations in understanding GPU performance metrics and scalability. It advocates for the use of proxy benchmarks to forecast the performance of arbitrary applications and discusses the considerations necessary for constructing future benchmark suites.

On the other hand, comprehensive stress testing software such as AIDA64 and HeavyLoad tests multiple components (CPU, GPU, RAM) simultaneously, providing a more holistic assessment of system performance and stability under extreme conditions. Although these tools help identify system bottlenecks and potential problems, configuring them and interpreting the results can be more time-consuming and effort-intensive compared to testing individual components. Automated stress testing tools like JMeter and LoadTracer simplify the stress testing process by providing detailed reports and analysis on system performance and stability. However, they may require more technical knowledge to install and configure [13, 19, 23].

Selecting stress testing software for a gaming computer should take into account specific needs, such as the level of detail required, the components to be tested, and the desired balance between comprehensiveness and ease of use. A combination of specialized and comprehensive tools can offer the most complete assessment of system performance and stability [13, 19, 23].

Pugh [15] underscores the challenges in selecting representative benchmarks and the complexities of evaluating computer system performance, which often surpass those in algorithm evaluation. Researches have address performance variability in computer systems for accurate assessments [9], benchmark energy efficiency, power costs, and carbon emissions in heterogeneous systems [11], compare different benchmarking software [5], develop strategies to enhance performance evaluation [14], and use of microbenchmarks to assess virtual machine instructions [3].

Figure 1 summarizes the main software commonly used for stress tests, detailing characteristics such as components tested, programming languages, algorithms, ownership, operating systems, and others. The illustration highlights that each tool offers different features, without a standardized approach. Each manufacturer provides their own benchmarking software, which varies in the parameters analyzed and the way results are presented. These tools provide machine characteristics and a reference score, which indicates the machine's suitability for gaming. The tools listed vary significantly in their focus, with some designed specifically for stress testing (e.g., FurMark for GPU stress, Prime95 used in HWMonitor for CPU stress) and others providing more general system monitoring and benchmarking (e.g., Geekbench, Novabench). The programming language also is also diverse, such as C++, OpenGL, and Vulkan, with specialized algorithms like Fur rendering and Anti-Aliasing used for stress testing. Most tools are compatible with Windows, with some supporting Linux and macOS, but there are gaps in cross-platform compatibility.

Also, the related works aforementioned discuss about different aspects used to assess stress in gaming computers. One of these features that deserves special attention is the degradation factor when a component is exposed to a high load, which is the stress test. When studying the degradation factors of the components, it became evident that these devices are composed of semiconductors [20], and that any degradation directly affects the performance of the components [6, 8].

Below is a list of the key factors which contribute to the degradation of components:

- Processing Demand: Programs or tasks requiring high processing power can subject computer components to more stressful conditions.

Info Stress Runtime Score Benchmark Windows Linux macOS Android

Software	Rev	Enterprise	CPU	GPU	RAM	SSD	FPS	OS	Programming	Github	Algorithm(s)	Support
FurMark	2	Geeks3D FR							OpenGL, Vulkan	✗	Fur rendering	Discord
CPU-Z	1	CPUID FR							MS Visual C++	anrieff BG	Prime95	Contact
HWMonitor	1	CPUID FR							MS Visual C++	✗	✗	Contact
GPU-Z	1	TechPowerUP							C++	✗	2D noise function	Forum
Geekbench	1	Primate Labs US							ANSI C, Lua	primatelabs	LZMA compression, Blowfish	Forum
SiSoftware Sandra	1	SiSoftware GB							OpenCL	✗	avx-iFMA, avx512-fp16, sha2-512	Ticket
SuperPosition	1	UNIGINE GB							C++/unigine	✗	SSRTGI (ray tracing)	Contact
Novabench	1	Novabench CA							?	✗	[compression and cryptography]	e-mail
MSI Kombustor	1	MSI TW							OpenGL, Vulkan	✗	Anti-Aliasing	e-mail
OCCT	1	OCBase FR							?	✗	3D Standard and Adaptative	Discord
3DMark	1	UL Solutions US						$	?	✗	For each game	Help
UserBenchmark	1	Free service							?	✗	?	Guide
Cinebench	1	Maxon DE							?	✗	Render across CPU/GPU	e-mail
CrystalDiskMark	1	CrystalMark JP							C++	hiyohiyo	Sequential and Random	e-mail
BaseMark GPU	1	BaseMark FI							C++/RockSolid	✗	State-of-the-art rendering	Contact
Google Benchmark Library	1	Google US							C++	google	✗	Forum

Figure 1: Benchmark review for different stress tests software and their main characteristics.

- Increased Temperature: Intensive processing generates more heat, especially in CPU and GPU, accelerating degradation processes like atom migration and material aging.
- Power Cycling: Frequent power on and off cycles can strain components, especially when temperature variations are significant.
- Overclocking: Increasing the CPU or GPU operating frequency beyond factory specifications can result in higher heat production and component stress, potentially accelerating degradation if not managed properly.
- Environmental Conditions: Factors such as high temperatures or humidity in the computer's environment can contribute to faster component degradation.
- Intensive Storage and Memory Usage: Activities involving frequent read/write operations or heavy memory use can stress components like memory controllers and disk controllers.

Figure 2 presents a flowchart showing the life-cycle of computer components from their initial phase to their eventual replacement or maintenance. It begins with components being integrated into the computer during the assembly process, followed by a phase of normal operation. Throughout this operation, factors such as natural wear and tear, thermal stress, and critical events (such as electrical overloads) can influence component degradation. These challenges result in component failures, leading to consequences such as processing errors, data loss, and a subsequent decline in performance.

The importance of environmental factors, such as temperature and humidity, which can accelerate semiconductor wear is also highlighted in Figure 2. The consequences of failures lead to decision-making, such as maintenance or replacement, and a bad experience for the player.

Most researchers argued that it is important to adopt practices of conscious use and ensure that the computer is maintained in suitable conditions to mitigate component degradation. It includes providing adequate ventilation, avoiding extreme environments, not overloading the system with intensive tasks, and if applicable, not overclocking without careful monitoring and proper cooling. Additionally, using temperature and performance monitoring software can help maintain control over the computer's operational conditions. Finally, it is worth noting that semiconductor degradation can indeed be analyzed in a laboratory setting through controlled experiments to verify changes in electrical and phonon conditions in the semiconductors.

In summary, this section has explored topics concerning performance evaluation, computer system benchmarks and pointed out the main tools and features. These studies underline the role of effective benchmarks in understanding both hardware and software performance, emphasizing the continuous need for enhancements and greater transparency in evaluation methods. This paper presents the CPU performance results of three gaming computers, as a white-box manner, which contributes to benchmark methods. These results are part of the development of software that will perform stress testing on the four components (CPU, GPU, RAM, SSD) and aims to be transparent in evaluating the machine's performance. An important feature of the algorithm presented in this work was the consecutive search of next N prime number without interruptions. It means that it performs a continue sequence of loops after find the next N prime, which differs to Prime95.

3 METHODOLOGY

This section details the methodology employed to develop and evaluate an algorithm based on prime numbers for stress testing CPUs, following the Goal, Question, Metric (GQM) approach [2].

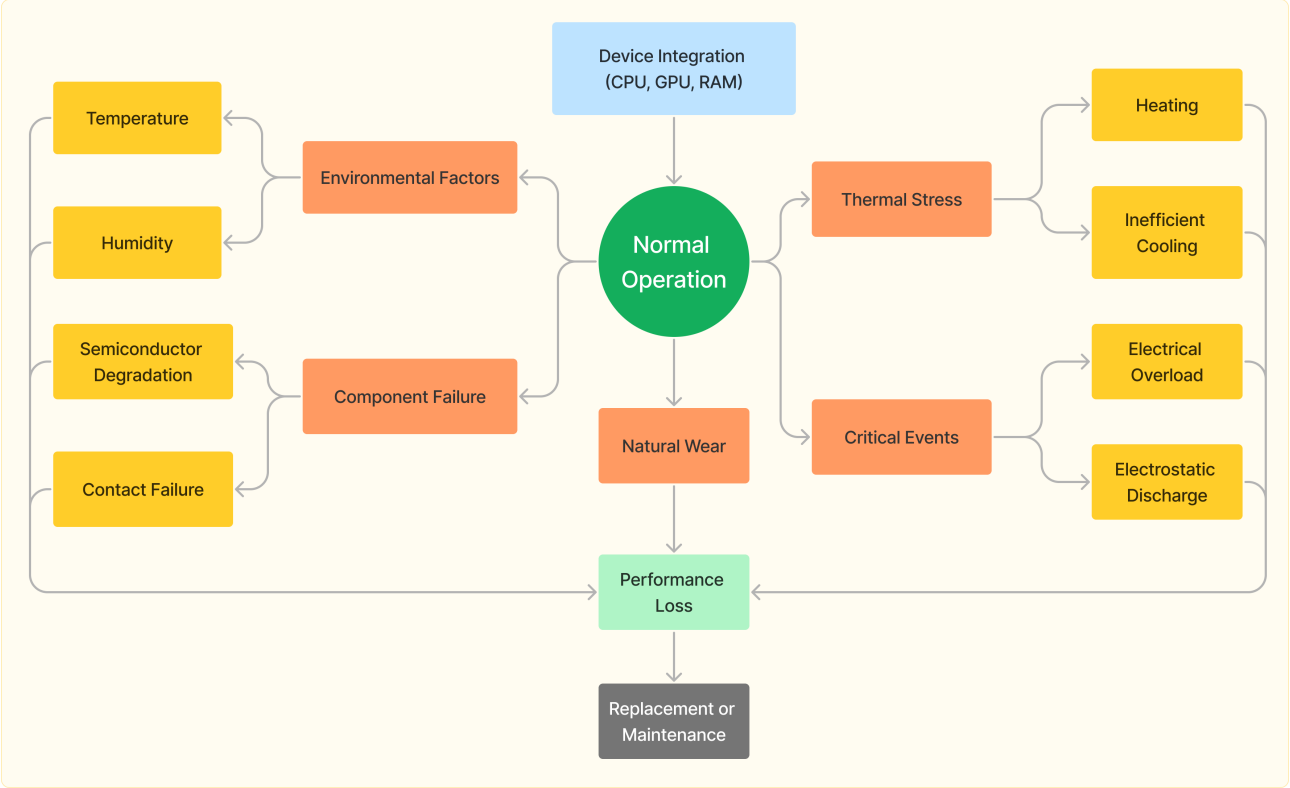

Figure 2: Degradation factors that can affect the performance of a gaming computer throughout its life cycle.

Goal: The primary objective is to develop an algorithm capable of quantifying CPU usage under stress conditions without causing long-term degradation.

Questions:

(1) What are the key characteristics of existing commercial benchmarking tools that can inform the design of the stress testing algorithm?
(2) Which factors contribute most to CPU component degradation, and how can the algorithm maximize CPU load while avoiding these issues?
(3) How does the developed algorithm perform across different CPU models in terms of utilization and stress?

Metrics:

- Characteristics of benchmarking tools: Identified from bibliographic research.
- CPU degradation factors: Derived from a detailed study on component wear and tear.
- CPU utilization under stress: Measured using the implemented algorithm based on prime numbers across three different gaming machines.

Process:

- Literature Review: The methodology began with a comprehensive bibliographic review of major commercial benchmarking tools, focusing on their characteristics and capabilities. This review informed the initial design of the stress testing algorithm.
- Component Analysis: We conducted a study to identify the primary factors leading to CPU degradation. These insights showed up the importance in designing an algorithm that maximizes processing load while ensuring component longevity.
- Algorithm Development: An initial version of the CPU stress algorithm was implemented in C++, targeting extreme CPU utilization while avoiding degradation. Through C++ and modern compilers, numerical and bitwise operations can be performed by addressing CPU cache memory, and so on, avoiding the use of the external components to the CPU that could bring on an extra runtime in stress algorithms.
  The algorithm was executed on three CPUs detailed in Table 1, under same conditions. These specific gaming laptops were selected by availability.
- Testing and Evaluation: The algorithm was executed several times (cycles) and in parallel on several previously configured threads. A flowchart of the algorithm's steps is shown in Figure 3. The stress tests involved running the algorithm

Table 1: Specification of each computer gamer used in the stress tests

ID	CPU	GPU	RAM	SSD
1. Nitro 5 AN515-44	AMD Ryzen 7 4800H	NVIDIA GeForce RTX 1650	8GB	512GB
2. Predator Triton 300 SE	12th Gen Intel(R) Core(TM) i7-12700H	NVIDIA GeForce RTX 3060	16GB	2TB
3. Predator Triton 500 SE	12th Gen Intel(R) Core(TM) i9-12900H	NVIDIA GeForce RTX 3080	32GB	1TB

in parallel with black-box software, capturing CPU activity, and performing re-engineering based on the data.

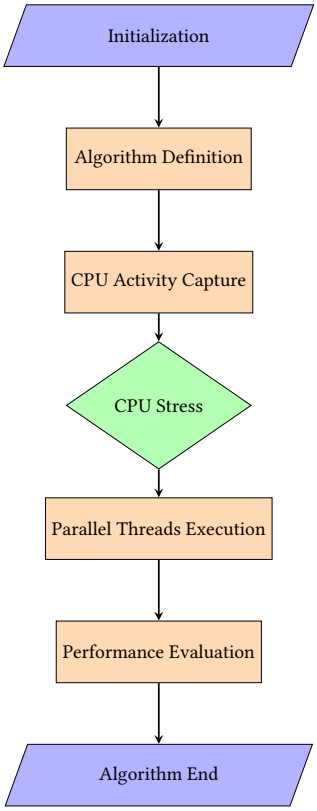

Figure 3: Description of the steps taken to build the CPU stress algorithm.

The algorithm was developed using the GetSystemTimes function (lpIdle, lpKernel, lpUser), where these variables represent the idle, kernel, and user process times, respectively. The stress test involves the calculation of prime numbers, whose function is responsible for finding the next prime number after a previously configured number, repeating this process a specific number of times. Essentially, the function seeks the next N prime numbers following H as detailed in Algorithm 1.

To determine the prime number, the smallest divisor of the integer is calculated, as shown in Algorithm 2. Therefore, during the stress test, the CPU is placed under heavy load by repeatedly performing prime number calculation in multiple threads, which requires high consumption of computational resources and results in a significant increase in CPU utilization.

Algorithm 1 NextPrimes function

```
1:  function NextPrimes(H, N)
2:      // Iterate N times to find N prime numbers after H.
3:      for k = 0 to N − 1 do
4:          while Divisor(H) ≠ H do
5:              H ← H + 1
6:          end while
7:      end for
8:      // Return the last prime number found.
9:      return H
10: end function
```

Algorithm 2 Divisor function

```
1:  function Divisor(H)
2:      // If H is less than 4, return 1, indicating it is prime or too
        small to have relevant divisors.
3:      if H < 4 then
4:          return 1
5:      end if
6:      // Iterate over possible divisors of H, starting from 2 up to
        H-1.
7:      for j = 2 to H − 1 do
8:          if H mod j = 0 then
9:              // If a divisor of H is found, return this divisor.
10:             return j
11:         end if
12:     end for
13:     // If no divisor is found, return H, indicating H is a prime
        number.
14:     return H
15: end function
```

The NextPrimes algorithm was chosen due to its ability to generate a sustained and significant computational load on the CPU. The task of finding successive prime numbers requires intensive arithmetic operations, particularly as numbers grow larger, which places a consistent demand on the CPU's processing capabilities. This algorithm's importance lies in its ability to simulate real-world scenarios where a system must handle complex, resource-heavy calculations, such as those found in cryptographic applications. Also, the algorithm is highly parallelizable, allowing the stress test to run across multiple threads simultaneously.

4 RESULTS

The C++ algorithm, which is based on prime numbers, was employed under the same conditions, to evaluate the CPU of three gaming computers previously described in Table 1. It can be used

for gaming industries, especially due to the organizational strategy, to evaluate gaming computers and offer this information to users.

The results of the stress tests are shown in Table 2. It is in terms of time spent to run the stress test (runtime). The algorithm was executed for 30 cycles and 17 threads (number of stress test instances that were executed simultaneously).

Table 2: CPU stress test

CPU	Runtime [sec]
1. Nitro 5 AN515-44	1905.981
2. Predator Triton 300 SE	926.432
3. Predator Triton 500 SE	568.004

The Predator Triton 500 SE presented the best performance, completing the test in 568.004 seconds in total. The Predator Triton 300 SE was placed in second with 926.432 seconds and Nitro 5 AN515-44 finished last with 1905.981 seconds. As all tests were run with 17 simultaneous threads, the difference in execution time directly reflects the processing capacity of each CPU. The algorithm showed stability, which means that the test measured performance without failures or errors. The analysis does not take into account the cooling characteristics of computers, which can affect performance in stress tests.

Similar results are seen for the tested machines when the performance of the proposed technique is compared with CPU-Z, which also uses prime number calculations; these findings are displayed in Table 3. This consistency proves that our proposed technique is appropriate for assessing microcomputer performance and fits in nicely with a popular benchmarking tool such as CPU-Z. It also shows that the method employed to capture the relative performance of these CPUs is successful.

Table 3: CPU stress test with CPU-Z

CPU	Score
1. Nitro 5 AN515-44	4727.5
2. Predator Triton 300 SE	4914.6
3. Predator Triton 500 SE	7275.0

Table 2 shows that Predator Triton 500 SE outperforms all tested processors under maximum load, as shown in Figure 4.

Based on the literature revised, we aggregate different protocols to analyze benchmarks for different computers. They are listed bellow.

(1) Identify performance limits and bottlenecks:
   - Observe the maximum CPU utilization levels reached during the stress test.
   - This will reveal the hardware components that are the limiting factors for your system's performance [12, 13].

(2) Assess system stability and reliability:
   - Monitor for any system crashes, freezes, or errors that occur during the stress test.
   - These indicate potential hardware or software issues that need to be addressed [1, 13].

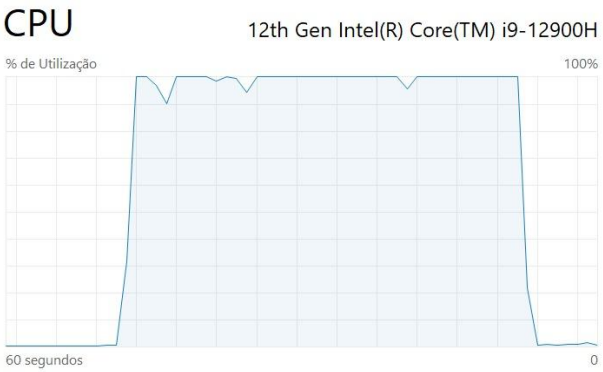

Figure 4: Stress test on the CPU of the Predator Triton 500 SE.

Through a stress test the users may have important insights in terms of performance, stability, and thermal management capabilities. This information can be used to ensure which part is suffering gaming workloads [1, 12, 13]. Also, running a benchmark is to stress the components of the machine to analyze their performance and check if they are capable of running games with good performance. This means imposing an overload on the components while ensuring that they are not degraded by use and, therefore, avoiding damage to the machine.

Finally, this study have implications in different domains, including for industry and academic sector. For practitioners and industry, the developed C++ algorithm offers a practical tool for diagnosing CPU performance under stress, being useful for hardware manufacturers, game developers, and IT professionals. For the academic community, it contributes to the ongoing research in performance evaluation and benchmarking.

4.1 Limitations and Future Works

The paper presents a simple and efficient C++ algorithm based on prime numbers to stress-test and measure the performance of three CPUs: Nitro 5 AN515-44, Predator Triton 300 SE, and Predator Triton 500 SE. This research is part of a larger software suite designed for stress testing CPUs, GPUs, RAM, and SSDs. As demonstrated in Section 2, several software tools are available for conducting CPU benchmarks.

We acknowledge threats to the validity of our findings. First, the specific hardware configurations used may limit the generalizability of the results to other systems with different specifications. Future studies should address these limitations by incorporating a broader range of tests, hardware configurations, and energy consumption metrics to provide a more comprehensive evaluation. Additionally, while the NextPrimes algorithm effectively stresses the CPU, it does not encompass the full range of workloads encountered in real-world applications, potentially limiting the breadth of our performance evaluations.

This study does not differentiate between single-core and multi-core performance. Single-core performance is particularly critical for most games, so it is essential for our tool to provide results for both scenarios. Future work may include the development of

a benchmarking framework that explicitly measures and reports both single-core and multi-core performance metrics. Furthermore, many games do not fully utilize the CPU's potential and run in parallel with other applications, such as voice programs. Thus, assessing the influence of these external factors on CPU performance will be valuable.

Energy consumption was not measured during the stress tests, although previous research has highlighted its significant impact, e.g. in cryptocurrency mining. This aspect shows further investigation in future studies.

Finally, we aim to apply the provided algorithm to a larger number of CPUs and conduct a comparative statistical analysis to enhance the robustness of our findings.

5 CONCLUSION

This paper identifies and details the main benchmark tools for stress testing computers. Each of them has its own characteristics and are based on different algorithm. However, many of them lack transparency in explaining how the score was calculated, despite often providing comparisons with other tested machines. Therefore, interpreting the results of stress tests can be somewhat complex.

From the user's perspective, it is important to check if the computer components are operating within safe temperature limits and are not subject to overload. Additionally, it is important to verify if the results fall within the reference limits for each test tool used and if the machine is suitable for gaming purposes.

Stress testing provides valuable information about the maximum performance of CPUs under extreme conditions. It is important to consider the user's specific context and needs when choosing a CPU. Factors such as price, energy consumption and portability must also be taken into consideration.

For users who prioritize maximum performance in games and demanding tasks, the Predator Triton 500 SE stands out as the best option. For users looking for a balance between performance and cost-benefit, the Predator Triton 300 SE could be a good choice. The Nitro 5 AN515-44 may be suitable for users on a more limited budget.

Our paper provides yet another method for testing the CPU, but this time using intuitive, proprietary code that prioritizes minimizing unnecessary load times. We get 100% CPU utilization with a fast test. Future works will be concentrated on developing a software that will perform stress testing on the four components (CPU, GPU, RAM, SSD) and aims to be transparent in evaluating the machine's performance.

REFERENCES

[1] Alim Adams. 2024. *Stress Test a Game: Four Key Considerations.* Antidote. antidote.gg/stress-test-game-four-key-considerations/ Accessed: 2024-07-05.

[2] Victor R Basili and David M Weiss. 1984. A methodology for collecting valid software engineering data. *IEEE Transactions on software engineering* 6 (1984), 728–738. https://doi.org/10.1109/TSE.1984.5010301

[3] Andreas Blumenthal, Mirko Luedde, Thomas Manzke, Bjoern Mielenhausen, and Christiaan E. Swanepoel. 2004. *Measuring software system performance using benchmarks.* United States, Patent Application Publication. Patent US2005120341A1_20050602. Online. Available at: searchplatform.rospatent.gov.ru/doc/US2005120341A1_20050602. Accessed: 2024-07-05.

[4] Shuai Che and Kevin Skadron. 2014. BenchFriend: Correlating the performance of GPU benchmarks. *Journal of Experimental Algorithmics* 28, 2 (2014), 238–250. https://doi.org/10.1177/1094342013507960

[5] Christopher Cullinan, Christopher Wyant, and Timothy Frattesi. 2012. *Computing Performance Benchmarks among CPU, GPU, and FPGA.* MathWorks, 117587, Russia, Moscow, Varshavskoye Highway, No. 125. Online. Available at: www.nicevt.ru/wp-content/uploads/2019/10/1.-Computing-Performance-Benchmarks-among-CPU-GPU-and-FPGA.pdf. Accessed: 2024-07-05.

[6] Tyler Dwyer, Alexandra Fedorova, Sergey Blagodurov, Mark Roth, Fabien Gaud, and Jian Pei. 2012. A practical method for estimating performance degradation on multicore processors, and its application to hpc workloads. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE, Salt Lake City Utah, 1–11. https://doi.org/10.1109/SC.2012.11

[7] Rudolf Eigenmann. 2000. *Performance evaluation and benchmarking with realistic applications.* MIT Press; First Edition, 255 Main Street 9th Floor Cambridge, MA 02142.

[8] David Flater and William F Guthrie. 2013. A case study of performance degradation attributable to run-time bounds checks on C++ vector access. *Journal of research of the National Institute of Standards and Technology* 118 (2013), 260. https://doi.org/10.6028/jres.118.012

[9] Samuel Irving, Bin Li, Shaoming Chen, Lu Peng, Weihua Zhang, and Lide Duan. 2020. Computer comparisons in the presence of performance variation. *Frontiers of Computer Science* 14, 1 (2020), 21–41. https://doi.org/10.1007/S11704-018-7319-2

[10] Desislava Ivanova, Vladimir Kadurin, and Yanko Belov. 2015. Performance Evaluation and Benchmarking of Modern GPU Architectures. In *International Scientific Conference Computer Science.* International Scientific Conference Computer Science'2015, Durrës, Albania, 1–8.

[11] Simon McIntosh-Smith, Terry Wilson, Amaurys Avila Ibarra, Jonathan Crisp, and Richard B. Sessions. 2012. Benchmarking Energy Efficiency, Power Costs and Carbon Emissions on Heterogeneous Systems. *Comput. J.* 55, 2 (2012), 192–205. https://doi.org/10.1093/comjnl/bxr091

[12] Samuel Nzube. 2024. *How to Stress Test GPU for Peak Performance.* Auslogics. www.auslogics.com/en/articles/how-to-stress-test-gpu-for-peak-performance/ Accessed: 2024-06-25.

[13] Samuel Nzube. 2024. *The Why and How of Computer Stress Tests: A User's Handbook.* Auslogics. www.auslogics.com/en/articles/the-why-and-how-of-computer-stress-tests-a-users-handbook/ Accessed: 2024-06-25.

[14] Aashish Shreedhar Phansalkar. 2006. *Measuring program similarity for efficient benchmarking and performance analysis of computer systems.* PhD thesis. University of Texas at Austin, Computer Science Dept. Taylor Hall 2.124 Austin, TXUnited States. ISBN: 978-0-549-26741-6.

[15] William Pugh. 2008. Technical perspective: A methodology for evaluating computer system performance. *Commun. ACM* 51, 8 (2008), 82–82. https://doi.org/10.1145/1378704.1378722

[16] Gareth Schott, David Buckingham, Andrew Burn, and Diane Carr. 2006. Studying computer games (Chapter 1). In *Computer Games: Text, Narrative and Play.* Willey, 42 McDougall Street Milton, Queensland 4064, 224pp. ISBN: 978-0-745-63400-5.

[17] Fadi N Sibai. 2007. Evaluating the CPU, Memory and Graphics Performance of Personal Computers with OSMark. In *Proceedings of the 9th Annual UAE University Research Conference.* College of Information Technology, United Arab Emirates University P.O. Box 15551, 1–6.

[18] Fadi N Sibai. 2008. Gauging the OpenSourceMark Benchmark in Measuring CPU Performance. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008).* IEEE, Institute of Electrical and Electronics Engineers, Portland, Oregon, USA, 433–438.

[19] TestSigma. 2024. *Software Stress Testing.* TestSigma. https://testsigma.com/blog/software-stress-testing/ Accessed: 2024-06-25.

[20] Vladislav A Vashchenko and Vladimir F Sinkevitch. 2008. *Physical limitations of semiconductor devices.* Vol. 340. Springer, New York, NY. 1–330 pages. https://doi.org/10.1007/978-0-387-74514-5

[21] Joao Victor Amorim Vieira, Matheus Alcântara Souza, and Henrique Cota de Freitas. 2023. Performance Evaluation of Intel and AMD Memory Hierarchies Using a Simulation-driven Approach With Gem5. In *Proceeding of XXIV Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD).* SBC, Brazilian Computer Society, Porto Alegre, RS, Brazil, 17–24. https://doi.org/10.5753/wscad_estendido.2023.235791

[22] Vijay. 2023. *18 Top Computer Stress Test Software To Test CPU, RAM And GPU [2023 LIST].* Software Testing Help. www.softwaretestinghelp.com/computer-stress-test-software/ Accessed: 2023-07-29.

[23] Vijay. 2024. *Computer Stress Test Software.* Software Testing Help. www.softwaretestinghelp.com/computer-stress-test-software/ Accessed: 2024-07-05.

Received 19 August 2024