



Benchmarking of Various Multicore Processors with Different Operating Systems

R. Kannadasan¹, A. S. Anakath²(✉), N. Prabakaran¹, A. Krishnamoorthy¹,
and S. Ambika²

¹ SCOPE, VIT University, Vellore, Tamilnadu, India

{kannadasan.r, krishnamoorthy.arasu}@vit.ac.in

² Department of Information Tecnology, E.G.S. Pillay Engineering College, Nagapattinam,
Tamilnadu, India

anakatharasan@gmail.com

Abstract. Benchmarking is the process of running programs and different heavy operations to measure the relative performance of various systems. With this project, we aim to design a tool and compare different systems with multicore processors and different operating systems. We will use various python libraries and user defined functions to perform demanding image processing and mathematical tasks to stress the systems. This can be used to get an idea of how well optimised the operating system is for the given hardware.

Keywords: Bench Marking · Multi Core Processors · Python Operating Systems and Libraries

1 Introduction

Several benchmarking applications are available online for benchmarking a system. One of the most popular benchmarking platforms is Geekbench. UserBenchMark is also another known platform. However, Geekbench is a synthetic benchmark and the results from the tests do not indicate real world performance of the system as such. Also, it is not open source, and there are companies that have exploited the lack of visibility of the code to make it boost the scores of their devices in the eyes of the customers. UserBenchMark has been known to be biased towards systems that feature components from Intel. Their scores also are largely centred around single core performance. Our aim is to create a simple benchmarking tool that assesses the performance of the processor (and the operating system) with the help of some tasks that stress the system. Both single core and multi-core intensive tasks will be present. It is also important that the tool is open source, so that the scores cannot be abused to cater to some company.

2 Literature Survey

We conducted a literature survey and analysed the results to get a better understanding on the current and past state of benchmarking applications and techniques. The brief summary of each of the papers are condensed into this literature survey.

[1] initially talks about how most benchmarks are weighted towards hardware and CPU performance, rather than the operating system the test is being run on. A good benchmarking program might be able to help developers get feedback about the changes they make without needing to actually get real feedback from users. [1] then discusses about some legacy benchmarks such as LMBench, UnixBench and HBBench-OS, which have not been updated since the 1990s. Some reasons are given for why it makes more sense to use a more modern type of benchmark, with examples of suitable candidates for the same. Finally, there is a discussion on all the performance information that an ideal benchmark should provide.

[2] explains the meaning and importance of the word “Benchmarking”, and answer why the research on benchmarking is still important and required in today’s modern era. [2] showcases a generic high level benchmarking architecture. The possibilities of different benchmarking software’s are depicted through “benchmarking cloud services” and “benchmarking cognitive radio solutions”. [2] clarifies the meaning and reason for benchmarking computers and computer networks in the domain of FIRE projects and highlights several benchmarking problems and challenges still to be looked into using the examples of “benchmarking application modelling” and “benchmarking cognitive radio benchmarks”.

[3] uses the standard benchmark known as wPrime to measure the multithreaded environment. wPrime benchmark suite allows the users to save the runtime score and sends it to its own website as well. The runtime scores are used for extra evaluation of the benchmark. Four different processors were compared in [3] Intel Core i3, Intel Core 2 Duo, Intel Pentium IV and the Asus atom processor. The runtime score under multithreaded benchmark is obtained. The results found showed that the performance of multi-core processors is superior in a multithreaded situation. The multi-core processors will give superior results as compared to the single core Asus atom or Intel Pentium IV processors. However, it was observed that when few threads were used the performance of the Intel Core 2 Duo was superior to the Intel Core i3 processor.

[4] compares the performance achieved and results of their standard benchmarks executed on selected architectures for various sets of data to identify all the possible bottlenecks. [4] discusses the most ideal practices and best suggestions for parallel software development to aid users to make a conclusion in advance and then select appropriate solutions to accelerate the execution of their applications. [4] analyses the optimized benchmarks of the 2013 CPU and GPU platforms using complex methods determined by the JPEG 2000 standard confirmed that, sometimes some parts adjust better to GPU architecture and other parts are efficiently executed on multi-core CPUs.

[5] describes the development of a set of benchmark programs which allows an individual to test and differentiate processors based upon the performance of the micro-architecture itself, independent of workload characteristics. The benchmark suite is comprehensive in its coverage of important hardware features that impact performance. The benchmarking programs are run directly on the hardware as opposed to processor simulators. The programs are designed to stress the system. The benchmarks were run on systems representing a different type of micro-architectures and developed an approach of the mathematical kind to analyse the results.

[6] discusses about the history of benchmarking tools in the context of evaluation of computer performance. Certain standard benchmark problems, synthetic jobs, workload mapping, application benchmarking as well as the advantages and disadvantages of various benchmarking techniques. The early benchmarking (Standard EDP Reports) is touched upon, with the early definition of benchmarks. They had some weaknesses such as the biases of the manufacturers claiming speeds of their products. Application benchmarks are discussed next. The proposed systems run the routines and there is consideration of the total throughput time. Synthetic programs attempt to mimic actual applications. Adjustable parameters can be used to account for changes in application and size. Finally, it moves on to the Standard Benchmark Library, which has the capability to simplify the evaluation of computer performance and provides a cost effective and reliable means to differentiate between the competition.

In [7], the AES algorithm was modified by removing the Mix Columns function and it was replaced with bit permutation because of its high computational requirement. The results of [7] show that the modified AES was able to increase the efficiency and it also had a greater avalanche effect. [7] compared the performance of the standard (Mix Column) and modified AES (Bit Permutation) algorithm by encrypting both text files and images. Both algorithms were evaluated based upon the encryption time, CPU usage and avalanche effect. [7] concluded that the modified AES has greater efficiency than that of the standard AES as it had a faster encryption time in text files as well as images.

[8] discusses the various techniques used in benchmarking in RTOS systems. The goal with these benchmarking techniques is to get the highest throughput and making sure the allocation is as fair as possible. There should not be a situation in RTOS where a lower priority task is executing when the task with a higher priority is waiting in the Queue. RTOS differs from general operating systems in that RTOS forces a strict priority of tasks. Higher priority tasks are given the ability to pre-empt a task with a lower priority that is currently executing. Advanced RTOS will be capable of high-speed processing of data. This will require high time bound processing, which is not really available currently. This shows the need to improve real-time mechanisms for the future.

[9] provides an insight into the benchmark development criteria as employed by the SPEC and TPC consortia. It provides a definition for benchmarks and rating tools, differentiating between benchmarks for competitive purposes and rating tools for research purposes, regulatory programs, or as part of a system improvement and development approach. [9] explains the differences between the three major types of benchmarks: specification-based, kit based, and hybrid. Finally, it describes the major quality criteria of industrial benchmarks: relevancy, repeatability, fairness, verifiability, and usability, including examples on how the criteria are ensured in standardized benchmarks.

[10] gives details on how to perform the Image processing tasks in Python programming language, so that it becomes easy for all to understand the concepts related to it. [10] also provides the use of Python Image Library (PIL), using which we can prominently develop the Python based image processing software and can be useful for number of applications like remote sensing, agriculture, space centre, satellites, medical and health sciences, etc. Thus, it can be concluded that Python and Image processing proves to be the better combination for learning, developing and understanding the capabilities

provided in it. It is also known that image processing tasks are an effective way to stress the system and compute its performance.

3 Proposed System

Our benchmark provides a set of simple tests that range from image processing tasks to mathematical tasks and finally ends with compression and encryption/decryption of images and text. It also uses Streamlit to provide a simple but effective interface to display the results from the tests. The time taken for each test as well as the total time for the test is displayed in the window. The image processing tasks involve blending and warping images, edge detection, gaussian blur, erosion and dilation. The mathematical tests involve matrix multiplication and a repeated factorial test. The last two tests are image and text encryption/decryption and image compression. There is specific emphasis on the difference between single core and multi core systems in some tests such as matrix multiplication, and the number of cores a system has is taken into consideration. When the test is run on the same system running different operating systems, the results vary, which shows how well optimised the operating system is for the given hardware. So this benchmark is not only a test for the hardware of the system, but also a test of how the operating system handles the tasks given to it. Tests like matrix multiplication of higher order matrices tend to utilize 100% of the CPU cores (Fig. 1).

4 Methodology

As the program is set to run, a series of tests are conducted which perform certain computations that are CPU intensive. These tests are specially designed to stress the system. The time for execution is saved and used as the means to compare the systems. The tests being used are:

1. BlendandWarp

In the blend and warp test, the program reads two pre loaded images. Then the image is blended and stored as a temporary file. This file is read again in gray scale and then it performs:

- a) Vertical wave
- b) Horizontal wave
- c) Concave effect
- d) Both vertical and horizontal wave

2. Edge Detection

In the edge detection test, the program loads up a set of images that are part of a separate folder. The program then uses algorithms to detect edges in the image (Fig. 2).

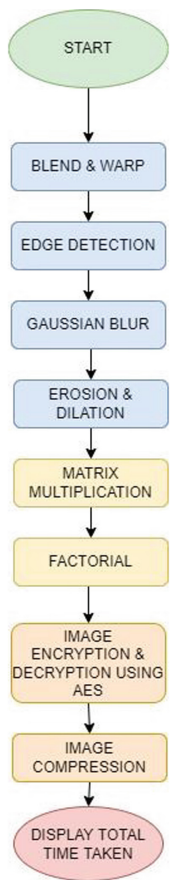


Fig. 1 Flow diagram of the proposed system



Fig. 2 Edge detection test



Fig. 3 Gaussian blur test

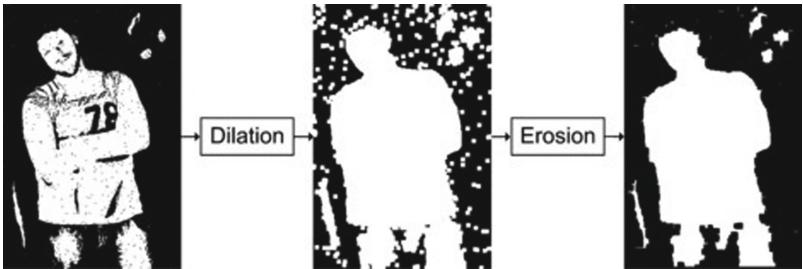


Fig. 4 Erosion and dilation test

3. Gaussian Blur

For the Gaussian Blur test, the program loads up a predefined image and performs Gaussian Blur on it. This is done through functions defined in the OpenCV library for python (Fig. 3).

4. Erosion and Dilation

For the erosion and dilation test, we once again make use of the OpenCV library which has the necessary functions to perform the test (Fig. 4).

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + cz + d \\ ex + fy + gz + h \\ ix + jy + kz + l \\ 1 \end{bmatrix}$$

Fig. 5 Matrix multiplication test

$$1! = 1$$

$$2! = 2(1) = 2$$

$$3! = 3(2)(1) = 6$$

$$4! = 4(3)(2)(1) = 24$$

$$5! = 5(4)(3)(2)(1) = 120$$

Fig. 6 Factorial calculations test

5. Matrix Multiplication

The multiplication test uses Python's multiprocessing library as well as NumPy to perform mathematical operations that stress all CPU cores at once to determine the multitasking capabilities of it. The multiplication is run several times as to increase the time for computations. A large matrix size has also been used to stress the system more (Fig. 5).

6. Factorial Calculations

The factorial test calculates the factorial of a large range of numbers. This test also creates large arrays and uses them to calculate the factorials. The factorial is calculated multiple times to increase execution time and stressing the system more (Fig. 6).

AES Encryption

In this test a sample text and image are encrypted and decrypted using a modified version of the Advanced Encryption Standard algorithm. The MixColumn function has been replaced by bit permutation. This test utilizes file handling to read sections of text and images to encrypt and decrypt them. It has high requirements and is able to stress the system (Fig. 7).

7. Image Compression

In this test, an image will be compressed multiple times. Python libraries will be used to implement this. It will stress the system making it ideal to be used in a benchmark tool (Fig. 8).

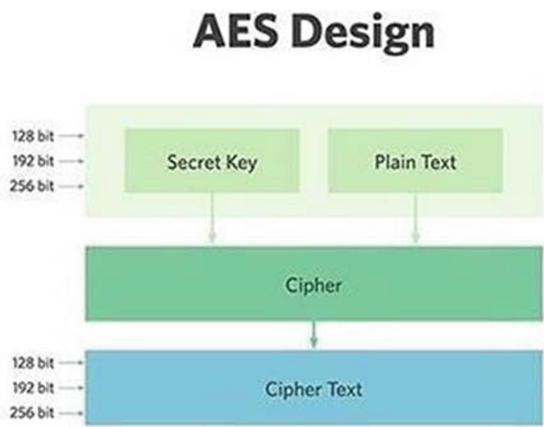


Fig. 7 AES encryption process

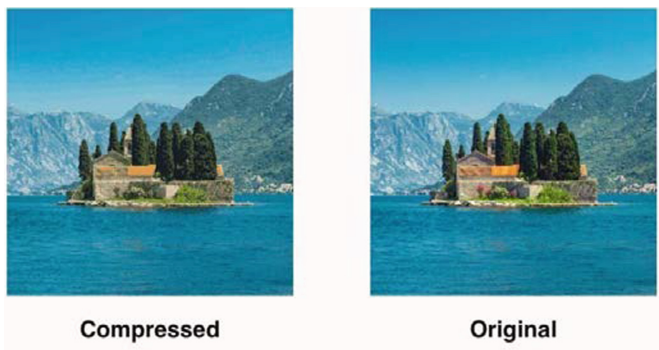


Fig. 8 Image Compression test

When each test is performed, the time taken to run the test is recorded and the score or total time is calculated. The sum of the scores of all the tests is saved and displayed to the user.

5 Results and Conclusion

The benchmark tool being developed has taken several ideas from the variety of literature papers surveyed. It will stress the CPU with different tasks such as blend and warp, edge detection, gaussian blur, erosion and dilation, matrix multiplication, factorial calculation, modified AES text and image encryption and decryption and image compression. Various CPUs with different operating systems are compared to determine which configuration performs better. It also helps illustrate how CPUs and operating systems have evolved with time and will keep developing in the future. Benchmarking tools will also need to keep up with the times and increase the complexity of the tests stressing the systems.

References

1. Hatt, Nicholas. "Benchmarking Operating Systems." (2008).
2. Bouckaert, Stefan & Vanhie-Van Gerwen, Jono & Moerman, Ingrid & Phillips, Stephen & Wilander, Jerker. (2011). BONFIRE: benchmarking computers and computer networks.
3. Madheswari, Neela & Banu, R.S.D.W.. (2011). Performance study of various processors using multithreaded benchmark suite. 58. 378–383.
4. Miłosz Ciżnicki, Michał Kierzyńska, Piotr Kopta, Krzysztof Kurowski, Paweł Gepner, Benchmarking Data and Compute Intensive Applications on Modern CPU and GPU Architectures, *Procedia Computer Science*, Volume 9, 2012, Pages 1900–1909, ISSN 1877–0509, <https://doi.org/10.1016/j.procs.2012.04.208>.
5. Deshmukh, Varad, Nishchay S. Mhatre and Shirang K. Karandikar. "Techniques for Benchmarking of CPU MicroArchitecture for Performance Evaluation." (2011).
6. Lewis, Byron C., and Albert E. Crews. "The Evolution of Benchmarking as a Computer Performance Evaluation Technique." *MIS Quarterly* 9, no. 1 (1985): 7–16. <https://doi.org/10.2307/249270>.
7. Gamido, Heidilyn & Sison, Ariel & Medina, Ruji. (2018). Modified AES for Text and Image Encryption. 11. 942–948. <https://doi.org/10.11591/ijeecs.v11.i3.pp942-948>.
8. Vetrivel, P., Kittappa Shanmuga Shanmuga and S. S. Krishnamurthy Babu. "A Survey of Benchmarking Techniques for Real-Time Operating System Performance Analysis." (2014).
9. von Kistowski, Jóakim & Arnold, Jeremy & Huppler, Karl & Lange, Klaus-Dieter & Henning, John & Cao, Paul. (2015). How to Build a Benchmark. ICPE 2015 - Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. <https://doi.org/10.1145/2668930.2688819>.
10. Harshada, Ms & Snehal, & Shitole, Sanjay & Pranaya, Mhatre & Suchita, Kadam & Shweta, Ghanate & Kurle, Darshana. (2016). Python Based Image Processing.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

