ISSN No:-2456-2165

Duplicate Image Detection and Comparison using Single Core, Multiprocessing, and Multithreading

[1] Pranav Kapoor, [2] Pratham Agrawal, [3] Zeeshan Ahmad, [4] Dr. Narayanamoorthi M

[1] Student, School of Computer Science & Engineering, VIT Vellore
2] Student, School of Computer Science & Engineering, VIT Vellore
[3] Student, School of Computer Science & Engineering, VIT Vellore
[4] Associate Professor Grade 1, School of Computer Science & Engineering, VIT Vellore

Abstract:- We undoubtedly have a chunk of images on our computer. The problem with having a lot of pictures is that you tend to accumulate duplicates along the way. It would be prudent to manage space efficiently. Detecting duplicate images from a set of images is a time-consuming task that can be automated, and duplicate data can be removed to save space. As we use our phones more, the number of unwanted duplicate photo and picture files grows in the device at random, ideally in every folder. Duplicate photos/pictures consume a lot of phone memory and slow down the phone's performance. Finding and removing them manually is difficult. Since human visual ability is not well developed enough to extract structure similarity from the naked eye, we propose a novel approach based on structural information degradation. As a practical solution to this problem, we create a structural similarity index and demonstrate it with a set of images from our database. Finding similar and duplicate photos from these samples can be a time-consuming task. Duplicate photo finders come in handy in this situation. Finally, we will compare the computation time and power required by processing on multiple cores vs. single core threads, as well as provide benchmarks and graphical representations for each.

Keywords:- *Single core ; Multithreading ; Multiprocessing; RGB ; Luminance ; Contrast ; Structure ; Similarity Index.*

I. INTRODUCTION

With the fast advancement of Internet technology and the growing use of technical devices, people may easily take, transfer, and share photos over the internet. Due to which duplicate pictures make up a substantial percentage of the image data. Duplicate image detection is the process of detecting duplicate copies of a query picture from a set of images efficiently and effectively.

Common devices like smartphones often duplicate images in form of junk files which is not supported by any security measures. These files are easily accessible to a non-authorized user and the user is unaware of such files. These files can contain important information which could be exposed against the user's permission.

To provide a proper and effective solution to this problem, we came up with measures that can detect and even eradicate such files with user permission.

The reason we are implementing it with multiple cores is that we want to compare it to sequential single-core execution and write about our findings.

By running the script and comparing using different cores, it will be possible to predict which solution is better for which type of operation.

For example, Single-core or sequential algorithm may be efficient for a small dataset consisting of just 5 images, as it may take up less CPU power, whereas when the dataset consists of 1000+ images, multi-processing might be efficient in detecting the duplicate images from the dataset.

II. LITERATURE REVIEW

[1] Proposes a method for evaluating perceptual picture quality that uses a range of known aspects of the human visual system to calculate the numeric visibility of mistakes (differences) between a deteriorated image and the original image. We offer an alternative complementary paradigm for quality evaluation based on the deterioration of structural information, based on the notion that human visual ability is highly adapted for obtaining structural information from a scene. We build a Structural Similarity Index as an illustration of this notion, demonstrating its promise with a set of intuitive examples and comparisons to both subjective evaluations and state-of-the-art objective approaches on a database of photos compressed using JPEG and JPEG2000.

[2] For picture registration, a similarity measure based on values from their respective Fourier Transforms is presented. The method generates signatures based on image content rather than image annotation and hence does not require human intervention. It computes the final rank for measuring similarity using both the real and complex components of the FFT. Any reliable method must precisely represent all objects in a picture, and different strategies may be required depending on the size of the image data. This paper gives an overview on how to use the Open-CV library to allow for the creation of a rating scheme for finding the similarity and further introducing a comparison metric based on the

intersection bounds of a covariance matrix of 2 images being compared with normalized Magnitude and Phase spectrum values. Using known methods of picture histogram comparison, sample findings on a test collection are supplied along with data. This strategy is particularly beneficial in photographs with variable degrees of brightness, according to the results.

[3] Introduces a method for edge identification in the image based on OpenCV with computer vision and image processing algorithms and algorithms for determining the exact number of copper cores in the microwire. To begin, we photograph the wire's interior structure with a high-resolution camera. Second, we implement picture pre-processing using OpenCV image processing methods. Finally, because morphological opening and shutting processes blur image borders, we employ them to segment images. Finally, contour tracking is used to determine the exact amount of copper cores. Experiments using Borland C++ Builder 6.0 reveal that OpenCV-based picture edge detection methods are straightforward, have a high level of code integration, and have a high level of image edge positioning accuracy.

[4] On astronomical data processing, proposes parallel processing techniques for multicore processors. Astronomers who prefer Python as their scripting language and utilized PyRAF/IRAF for data processing are the target audience. Three issues of varying difficulty were analyzed on three distinct types of multicore processors to show the benefits of undergoing parallelizing data processing activities in terms of execution time. The parallel code can be implemented rather easily thanks to Python's native multiprocessing module. The 3 multiprocessing approaches—Pool/Map, Process/Queue, and Parallel Python—were also compared.

[5] The paper states about the popularity of Python among numeric groups, because of its easy-to-use number-crunching modules such as [NumPy], [SciPy], [Dask], [Numba], and others.
To use all of the available CPU cores, these modules often use multi-threading for efficient multi-core parallelism. However, when utilized jointly in a single application, their threads can interfere with one another, resulting in overhead and inefficiency.

[6] Presents a revolutionary robotics middleware and programming environment that is deeply entrenched. It runs a multithreaded, publish-subscribe design paradigm for microcontroller applications and provides a Unix-like software interface. By giving a modular and standards-oriented platform, we improve on the embedded open-source systems. The system architecture is based on a POSIX application programming interface and a publish-subscribe object request broker.

[7] Proposes a framework for Image inpainting is a technique for guessing missing pixel values in an image by combining pixel value information from nearby pixels with prior knowledge gained through learning the object class. We present an agile image inpainting method that is very accurate and is based on subspace similarity in this work. In the learning phase, the suggested method generates the subspace from a chunk of images related to the object class, then estimates the null or missing pixel values of the provided image belonging to the same object class in the inpainting step to maximize the similarity between the image which is given as input and the subspace.

[8] Gives specifics of picture capture, address segmentation, and recognition, which are crucial techniques in automatic letter sorting machines, as well as the applications of pattern recognition technology to postal automation in China. Letter sorting machines with pattern recognition technology are frequently employed in mail processing centers to mechanically sort mail. Since the 1990s, approximately 100 letter sorting devices have been implemented throughout China. Siemens (previously AEG), NEC, Solystic (formerly Actel-Bell), and SRI (Shanghai Research Institute of Postal Science, China Post Group) were among the companies that produced these machines.

[9] Proposes an approach for extracting distinguishing invariant elements from photos that can be used to match multiple views of the same object or scene reliably. The features are invariant to the rotation of the image and its scaling, and they've been shown to give reliable matching across a long range of affine distortion, 3D perspective change, noise addition, and lighting change. The features are extremely distinctive in the sense that a single feature may be accurately matched against a vast database of features from multiple photos with a high likelihood. It also explains how to apply these traits to object recognition. Individual features are matched to a collection of features from known objects using a fast nearest-neighbor technique, then a Hough transform is used to identify clusters belonging to a single item, and finally, a least-squares solution is used to verify consistent posture parameters.

[10] Presents an area-level visual consistency verification scheme for partial-duplicate search to determine whether there is visually consistent region (VCR) pairs between images. The possible VCRs are created by mapping the regions divided from candidate pictures to a query image using the attributes of the matched local features to handle the issue of identifying partial-duplicate images from non-partial-duplicate images following the local feature matching.

III. ALGORITHM USED

➢ **Serial execution Algorithm**

A sequential algorithm, also named a serial algorithm, is a computer program that is run sequentially, rather than concurrently or in parallel, from beginning to end.

Most conventional computer algorithms are sequential algorithms, even if they aren't explicitly labeled as such because sequential is a background assumption.

Concurrency and parallelism are two different notions in general, although they frequently overlap (many distributed algorithms are both concurrent and parallel), hence the term

"sequential" is used to contrast the two without specifying which is which. If you need to distinguish between them, utilize the opposing pairings sequential/concurrent and serial/parallel. The term "sequential algorithm" can also apply to a decoding technique for a convolutional code.

➢ Parallel execution Algorithm

In parallel programming, the divide and conquer technique is applied. Separate and Conquer This technique can be broken down into three sections:

Divide: This entails breaking the problem down into smaller chunks.

Conquer: Call the subproblem recursively until the subproblem is solved.

Combine: The Subproblem is Solved so that we may find an answer to the problem.

Recurrence Relation - This is recurrence relation for above program.

```
O(1)  if n is small
T(n) =     f1(n) + 2T(n/2) + f2(n)
```

Divide and Conquer algorithm:

```
DAC(a, i, j)
{
    if(small(a, i, j))
      return(Solution(a, i, j))
    else
      m = divide(a, i, j)          // f1(n)
      b = DAC(a, i, mid)           // T(n/2)
      c = DAC(a, mid+1, j)         // T(n/2)
      d = combine(b, c)            // f2(n)
   return(d)
}
```

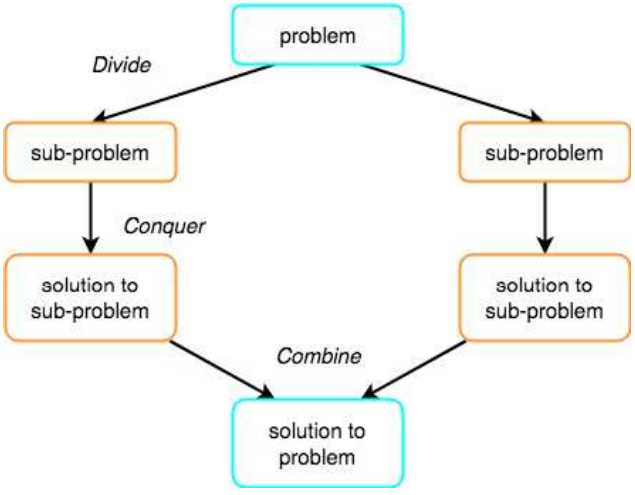

Fig1-Source:https://www.studytonight.com/data-structures/merge-sort

IV. PLATFORMS USED

• PyCharm

PyCharm is an integrated development atmosphere (IDE) utilized in Computers for programming, specifically for the Python language. it's developed by the Czech company JetBrains. It helps in tasks such as code analysis, a graphical computer program, an integrated unit tester, and further also supports internet development with Django likewise as knowledge Science with boa. PyCharm is cross-platform, with Windows, macOS, and UNIX operating system versions.

• Visual Studio Code

Visual Studio Code is a free source-code editor created by Microsoft for Windows, UNIX system, and macOS. options embrace support for debugging, syntax lightness, intelligent code completion, snippets, code refactoring, and embedded stinker. Users can make amendments to the theme, keyboard shortcuts, preferences, and install extensions that add further practicality.

• OpenCV

OpenCV (Open source Computer Vision Library) is a library of programming functions principally geared toward period laptop vision. The library is cross-platform and free to be used underneath the ASCII text file BSD license.

• Multiprocessing

Multiprocessing may be a package that supports spawning processes using an API the same as the threading module. Due to this, the parallel processing module permits the technologist to fully leverage multiple processors on a given machine. It runs on each UNIX as well as Windows. The parallel processing module conjointly introduces APIs that don't have analogs within the threading module.

• Threading

This module constructs higher-level threading interfaces on high of the lower level _thread module. This module provides low-level primitives for operating with multiple threads (also referred to as lightweight processes or tasks) — multiple threads of management sharing their world knowledge area. For synchronization, easy locks (also referred to as mutexes or binary semaphores) are provided. The threading module provides a better-to-use and higher-level threading API designed on high of this module.

V. PROPOSED APPROACH

We have created an operating Script to observe duplicate pictures that identify duplicate pictures recursively using the data multiprocessing library of python. To cipher the distinction between 2 pictures we'll be utilizing the Structural Similarity Index, 1st introduced by Wang et al. in their 2004 paper, Image Quality Assessment: From Error Visibility to Structural Similarity. This technique is already enforced within the scikit-image library for the image process.

The trick is to determine how we will verify specifically wherever, in terms of (x, y)- coordinate location, the image

variations occur in the area unit.

In a comparison of every image, we are going to notice the structural similarity between every image, and therefore the formula will observe the image with 100% similarity and take into account it as a duplicate or a copy.

To accomplish this, we'll first have to be compelled to check that our system has Python, OpenCV, scikit-image, and imutils.

Serial formula Vs Parallel formula

1. Sequential –
• Compare 2 pictures by their RGB values and notice image similarity victimization cv2 library consecutive.

2. Multiple-Processors
• Compare 2 pictures by their RGB values and notice image similarity victimization cv2 library by spawning processes victimization data processing library API.

3. Threading –
• By forwarding each comparison to a different thread, you may compare two images based on their RGB values and detect image similarity using the cv2 library.

Multithreading on the C-Python interpreter does not support full multi-core execution. Python does, however, include a Threading library.

So, what's the point of utilizing the library if we can't (allegedly) leverage many cores? Many programs, especially those involving network programming or data input/output (I/O), are network- or I/O-bound.

This indicates that the Python interpreter is waiting for the outcome of a function call that manipulates data from a "remote" source, such as a network address or a hard disk.

Reading from local memory or a CPU cache is far faster than this.

As a result, if several data sources are being visited, one way to speed up such code is to create a thread for each data item that needs to be retrieved.

Consider a Python script that scrapes a large number of web URLs. Given that each URL will have a download time that exceeds the computer's CPU processing power, a single-threaded approach will be heavily I/O bound.

Time of execution and CPU consumption for sequential, multi-processing, and multi-threading are compared and visualization is created by examining and evaluating all of the above ways.

The script can install multiple data sources in parallel and collect the results after each installation by creating a new thread for each installed resource.

This means that each subsequent download is not waiting on the download of earlier web pages. In this case, the program is now bound by the bandwidth limitations of the client/server(s) instead. The same is the case for searching an image directory.

For Smaller Sample Thread works better but on the creation of too many threads then comes context switch in play which acts as a barrier.

But Creating a Thread is a cheap process when creating a limited number of threads as they all are sharing the same resources and hence quite faster than Multiple Processing at that time.

Multiple Processors have their resource bucket, they are GIL safe and can work parallelly because of this reason with an increase in data they worked better than threading.

VI. ARCHITECTURE / BLOCK DIAGRAM

Most Image quality assessment techniques rely on quantifying errors between a reference and a sample image.

- A common metric is to quantify the difference in the values of each of the corresponding pixels between the sample and the reference images (By using, for example, Mean Squared Error).
- The Human visual system is highly capable of identifying structural knowledge from a scene and hence identifying the minute and minuscule differences between the information extracted from a reference and a sample scene. Hence, a measurement that mirrors this behavior will perform better on tasks that involve finding the difference between a sample and a reference image.

The Structural Similarity Index (SSIM) metric extracts 3 key features from an image:

1. **Luminance**
2. **Contrast**
3. **Structure**

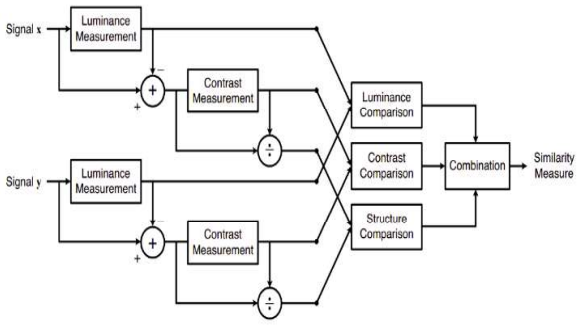

Fig 2: The Structural Similarity Measurement System. Source: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

The comparison between the two images is performed based on these 3 features.

This system calculates the Structural Similarity Index between 2 given images which is a value between -1 and +1.

ISSN No:-2456-2165

A value of +1 indicates that the 2 given images are very similar or the same while a value of -1 indicates the 2 given images are very different. Often these values are adjusted to be in the range [0, 1], where the extremes hold the same meaning.

Luminance: Luminance is measured by averaging over all the pixel values. Its denoted by μ (Mu) and the formula isgiven below

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i.$$

Fig 3: Where xi is the ith pixel value of the image x. N is the total number of pixel values. Source:https://www.cns.nyu.edu/pub/eer /wang03-reprint.pdf

Contrast: It is measured by taking the standard deviation (square root of variance) of all the pixel values. It is denoted by σ (sigma) and represented by the formula below,

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)^2\right)^{\frac{1}{2}}.$$

Fig4-Source: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

Structure: The structural comparison is performed using a consolidated formula (more on that later), but in essence, we divide the input signal by its standard deviation, resulting in a result with unit standard deviation, allowing for a more robust comparison.

$$(\mathbf{x}-\bar{\mu}_x)/\sigma_x$$

where x is the Input Image

Fig5-Source: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

VII. RESULT

DETECT DUPLICATE IMAGES VIA SINGLE AND MULTI-PROCESSING

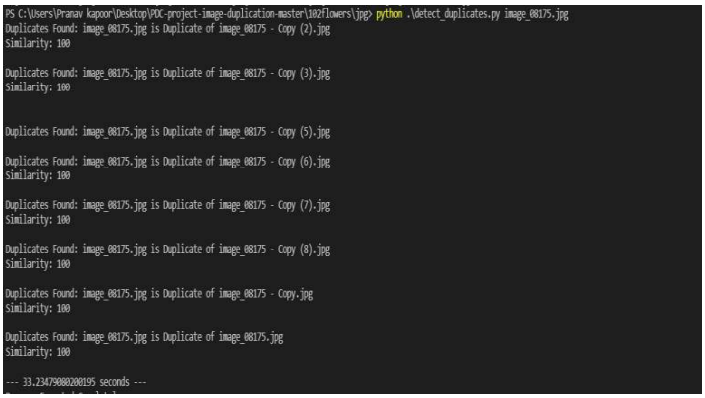

Fig 6 Generating duplicate images and the similarity percentage

Time is taken for single-core execution: -- 33.2 sec

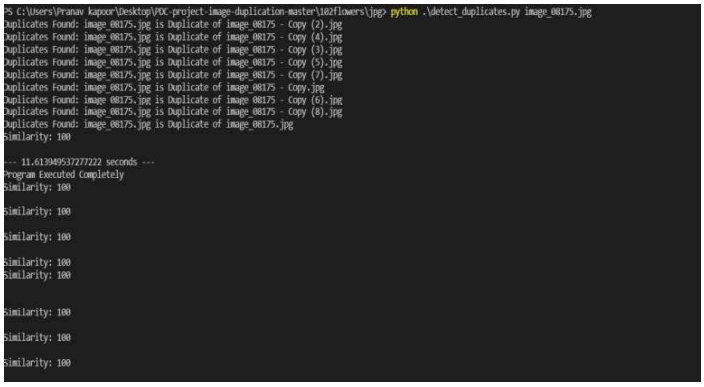

Fig 7 Generating duplicate images and the similarity percentage

Time is taken for multi-process and multi-thread execution: --11.6 sec which is far less than that of single-core execution

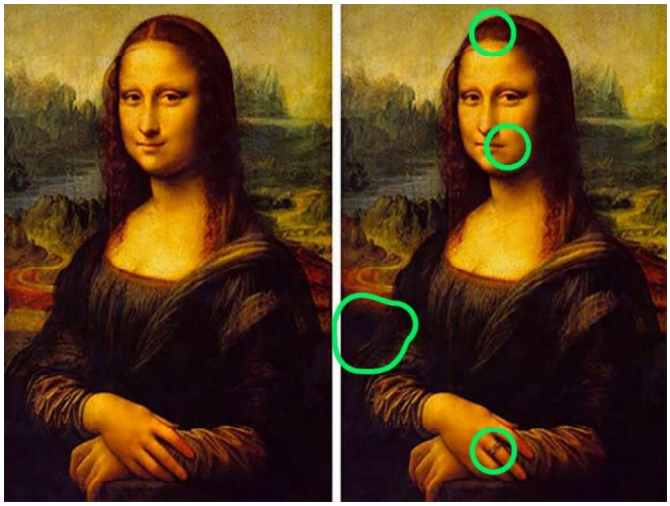

Fig8 : Source:https://www.jesusdaily.com/can-you-spot-the-difference-in-these-two-mona-lisa-pictures-there-are-4-total

CPU USAGE

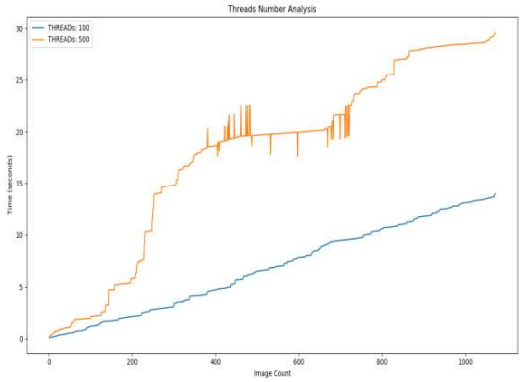

Fig 9: Thread Number analysis where the x-axis is Image count and the y-axis is Time in seconds

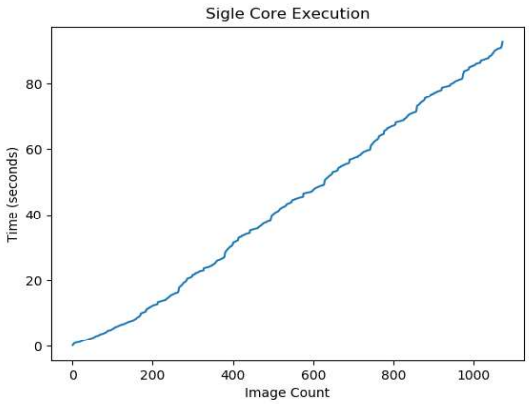

Fig 10: Duplicate detection in single-core execution wherethe x-axis is Image count and the y-axis is Time in seconds

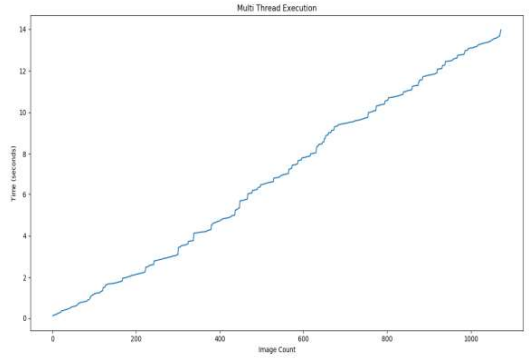

Fig 11: Duplicate detection in multi-thread execution where the x-axis is Image count and the y-axis is Time in seconds

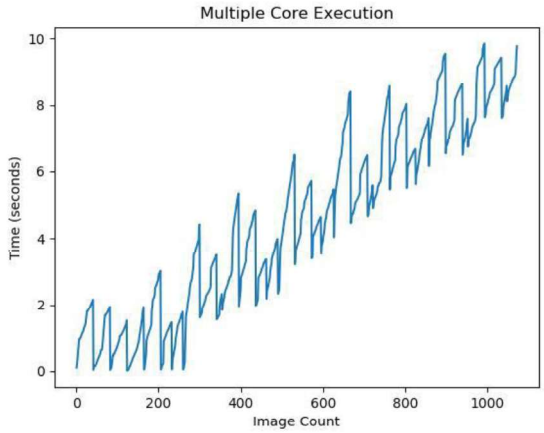

Fig 12: Duplicate detection in multi-thread execution where the x-axis is Image count and the y-axis is Time in seconds

Here we are having this pattern because each core had an equal amount of work divided, now they all merging their chunked worked output which is creating this wave-like pattern.

It is observable this line graph can be broken down into chunks of the group of 8 cores generating output together for their particular work.

We can clearly observe here that the Time Taken for Single Core, Single Threaded applications increase exponentially with Time.

But for Multiprocessing and Multithreading, it increases with a little slope value.

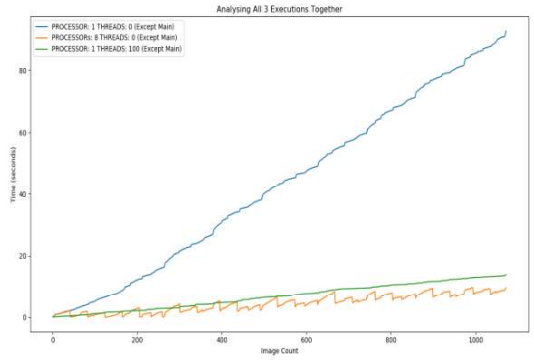

Fig 13: Duplicate detection in all 3 execution where the x-axis is Image count and the y-axis is Time in seconds

An addition to the observation is the fact that Multiprocessing Library is beating the thread as data is increased this is because Multiprocessing is resource extensive and so takes higher time to start its processing but then it processes the data like a charm whereas Thread have interrupts which are continuously degrading the performance of the application and it is not able to perform any better.

Here we see that serial execution of the processes takes up exponential time to process the code while on the other hand multithreading and multiprocessing both take up very little time for the execution.

Another thing that we found is that multiprocessing works slowly at first because the generation of threads is faster than the generation of processors but later on since every processor has its own GIL so parallelization is observed in a real-time scenario.

HOW IS CPU INFLUENCED?

- CPU Utilization becomes 88% for both Multiprocessing and Multithreading (depends upon the Hardware sometimes a 100%) .
- CPU utilization remains lower and around 26% which is normal for Single Processing and Single-Threaded application.

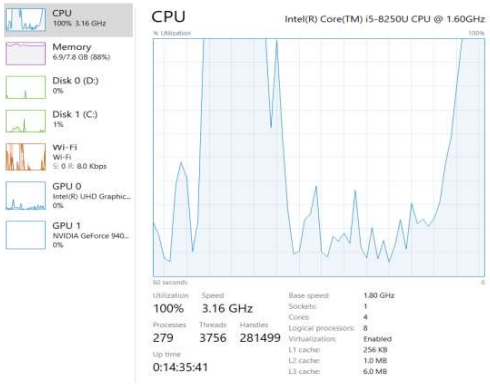

Fig 14: Analyzing the CPU usage in MultiProcessing

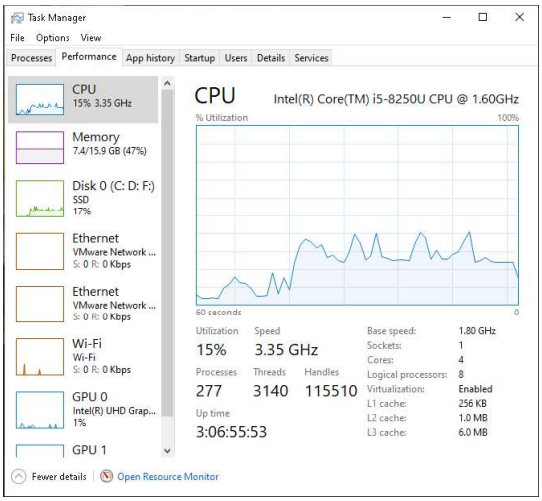

Fig 15: Analyzing the CPU usage in Single core Processing

VIII. CONCLUSION

The use of Threading is enough and more suited in Applications like GUI and Networking as it's less resource-intensive, Both Multiprocessing and Multithreading have similar performance in I/O bound operations.

Since most online servers provide single-core only so threaded apps are more used but for Research purposes, we have needed more computation power and parallel execution with less complication, and there comes Multiple Processors in to rescue.

IX. REFERENCES

[1] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.
[2] Narayanan, S., & Thirivikraman, P. K. (2015). Image similarity using Fourier transform. Journal Impact Factor, 6(2), 29-37.
[3] Xie, G., & Lu, W. (2013). Image Edge Detection Based On Opencv. International Journal of Electronics and Electrical Engineering, 1(2), 104-6.
[4] Singh, N., Browne, L. M., & Butler, R. (2013). Parallel astronomical data processing with Python: Recipes for multicore machines. Astronomy and Computing, 2, 1 10.
[5] Malakhov, A. (2016, July). Composable multi-threading for Python libraries. In Proceedings of the Python in Science Conferences.
[6] Meier, L., Honegger, D., & Pollefeys, M. (2015, May). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In 2015 IEEE international conference on robotics and automation (ICRA) (pp. 6235-6240). IEEE.
[7] "Fast image inpainting using similarity of subspace method" the authors Hosoi, T., Kobayashi, K., Ito, K.,

& Aoki, T.
[8] Y. Lu, X. Tu, S. Lu and P. S. P. Wang, "Application of pattern recognition technology to postal automation in China" in Pattern Recognition and Machine Vision-in Honor and Memory of Professor King-Sun Fu., Copenhagen, Denmark:River Pub. Co, pp. 367-381, Mar. 2010.
[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", Int. J. Comput. Vis., vol. 60, no. 2, pp. 91-110, 2004.
[10] Zhili Zhou, Q. M. Jonathan Wu, Yimin Yang, and Xingming Sun. 2020. Region-Level Visual Consistency Verification for Large-Scale Partial-Duplicate Image Search. ACM Trans. Multimedia Comput. Commun. Appl. 16, 2, Article 54 (June 2020), 25 pages. DOI:https://doi.org/10.1145/3383582.
[11] A. Landge and P. Mane, "Near duplicate image matching techniques," 2016 International Conference on Information Communication and Embedded Systems (ICICES), 2016, pp. 1-5, doi: 10.1109/ICICES.2016.7518863.
[12] K K, Thyagharajan & Kalaiarasi, G.. (2020). A Review on Near-Duplicate Detection of Images using Computer Vision Techniques. Archives of Computational Methods in Engineering. 28. 10.1007/s11831-020-09400-w.
[13] Kumar, P.J. & Ellappan, V. & Badala, P.. (2016). Image duplication detection. International Journal of Pharmacy and Technology. 8. 25632-25639.
[14] Morra, Lia & Lamberti, F.. (2019). Benchmarking unsupervised near-duplicate image detection.
[15] S. Thaiyalnayaki, J. Sasikala, R. Ponraj,Indexing near-duplicate images in web search using minhash algorithm,Materials Today: Proceedings,Volume 5, Issue 1, Part 1,2018,Pages 1943-1949,ISSN 2214-7853,https://doi.org/10.1016/j.matpr.2017.11.297.
[16] Malakhov, Anton & Liu, David & Gorshkov, Anton & Wilmarth, Terry. (2018). Composable Multi-Threading and Multi-Processing for Numeric Libraries. 18-24. 10.25080/Majora-4af1f417-003.
[17] Walter Tichy, "The Multicore Transformation", Ubiquity, Volume 2014 Issue May, May 2014. DOI: 10.1145/2618393.http://ubiquity.acm.org/article.cfm?id=2618393
[18] Maret, Yannick. (2007). Efficient image duplicate detection based on image analysis. 10.5075/epfl-thesis-3797.
[19] Y. Cao, H. Zhang, Yanyan Gao and Jun Guo, "An efficient duplicate image detection method based on Affine-SIFT feature," 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT),2010,pp.794-797,doi: 10.1109/ICBNMT.2010.5705199.
[20] Z. Zhou, Q. M. J. Wu, S. Wan, W. Sun and X. Sun, "Integrating SIFT and CNN Feature Matching for Partial-Duplicate Image Detection," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 4, no. 5,pp.593-604,Oct. 2020, doi: 10.1109/TETCI.2019.2909936.