ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

BENCHMARKING PARA PROCESAMIENTO PARALELO MEDIDA DEL DESEMPEÑO DE LA EJECUCIÓN PARALELA UTILIZANDO PYTHON

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

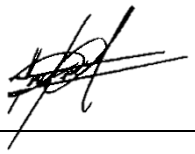
PAUL STALIN GUAMAN MOROCHO
paul.guaman@epn.edu.ec

DIRECTOR: RAMIRO MOREJÓN
ramiro.morejon@epn.edu.ec

DMQ, Febrero 2023

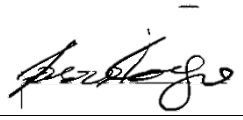
CERTIFICACIONES

Yo, PAUL STALIN GUAMAN MOROCHO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



PAUL GUAMAN

Certifico que el presente trabajo de integración curricular fue desarrollado por PAUL STALIN GUAMAN MOROCHO, bajo mi supervisión.



RAMIRO MOREJÓN
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

PAUL GUAMAN

RAMIRO E. MOREJÓN T.

DEDICATORIA

A mis padres que gran amor e inculcandome los caminos del cristianismo me han permitido alcanzar todos mis objetivos incluido este.

A mis hermanos que han sido el apoyo incondicional en los tiempos más complejos.

A mi abuelita Juana que con su infinito amor, consejos y sabiduria me ha ayudado a seguir adelante.

Paúl

AGRADECIMIENTO

A Dios, por darme la fortaleza espiritual de afrontar cada una de las dificultades que se presentaron en el largo proceso del estudio universitario.

A mi madre, que con sencilles y humildad, ha estado en cada momento de mi vida con su apoyo incondicional, aun en los momentos más difíciles, mostrandome la gran fortaleza con la que se deben afrontar cada una de las situaciones de la vida, siempre estando pendiente para lograr todos los objetivos que he alcanzado a lo largo de mi vida.

A mi padre, que me ha brindado su apoyo en cada uno de los ámbitos que he necesitado, me ha brindado todos los medios necesarios para logras las metas que me he propuesto, su amabilidad y respeto para tratar a los demás y su dedicación en cada trabajo, han sido las mayores enseñanzas que he podido aprender y enseñandome a nunca dejar un sueño y simpre luchar por lograr algo mejor.

A mi hermano German, que siempre me brindo su apoyo en cada una de las situaciones, aún en las más difíciles, es mi gran ejemplo de superación y dedicación, a pesar de cualquier dificultad, que me han servido para alcanzar esta gran meta.

A mi hermano Darwin, el cual a más de hermano es el mejor amigo que tenido a lo largo de mi vida, con quien he superado grandes dificultades, ha estado conmigo en cada momento apoyandome para lorar conseguir mi mayor objetivo, terminar mis estudios universitarios.

Finalmente, a la Escuela Politécnica Nacional y al Ing. Ramiro Morejón por la acertada directriz y soporte durante el desarrollo del presente proyecto permitiéndome finalizar mi etapa en esta prestigiosa universidad de la mejor forma posible.

ÍNDICE DE CONTENIDO

RESUMEN

Este trabajo integrador trata sobre la medición del desempeño de la ejecución paralela con la utilización de Python. Este componente se centra en la addquiición de tiempos de ejecución lineal y paralelo de funciones matemáticas, las cuales se describiran en este documento. Para este propósito se utilizó el lenguaje de programación Python, el cual permite a implementación de la ejecución paralela.

Se presentan conceptos referentes a la ejecución paralela, procesamiento multi-hilo (Threading) y multiprocesamiento (Pool). También se presenta el algoritmo implementado para la obtención de los tiempos de ejecución de las funciones implementadas.

Con el fin de realizar la adquisición de los tiempos de ejecución, se definió un intervalo de confianza para cada una de las funciones implementadas en el presente trabajo.

PALABRAS CLAVE: Python, Threading, Pool, Intervalo de confianza.

ABSTRACT

This final degree project deals with measuring the performance of parallel execution using Python. This component focuses on the acquisition of linear and parallel execution times of mathematical functions, which will be described in this document. For this purpose, the Python programming language was used, which allows the implementation of parallel execution.

Concepts related to parallel execution, multi-thread processing (Threading) and multi-processing (Pool) are presented. The algorithm implemented to obtain the execution times of the implemented functions is also presented.

In order to carry out the acquisition of execution times, a confidence interval was defined for each of the functions implemented in this work.

KEYWORDS: Python, Threading, Pool, Confidence Intervals.

1. INTRODUCCIÓN

DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El procesamiento paralelo es una procpiedad con la que los proccesadores cuentan en la actualidad, este metodo de ejecución es ampliamente utilizado en pruebas de benchmark para procesadores que cuentan con multiples núcleos, actualmente los procesadores desarrollados cuentan con la capacidad multinúcleo.

Uno de los lenguajes de programación que nos permite la implementación de la ejecución paralela es Python, sobre el cual el presente trabajo se desarrolla.

La característica multiplataforma de Python posibilita la implementación del procesamiento paralelo sobre varias plataformas (sistemas operativos), así como también sobre equipos los cuales poseen distintas características de hardware.

La plataforma Python permite la implementación de ejecución paralela, versiones anteriores de la plataforma simplemente permite la ejecución multi-hilo(Threading), la versión más reciente de la plataforma Python proporciona la ventaja de trabajar con multiprocesamiento gracias al módulo multiprocessing y los múltiples objetos que dispone dicho módulo.

La ejecución multi-hilo presenta una serie de restricciones para la implementación de la ejecución paralela, por lo cual la utilización del módulo Threading presenta dificultades para el trabajo desarrollado. Por el contrario, el módulo multiprocessing ha sido desarrollado con el propósito de eliminar las restricciones que su antecesor el módulo Threading, el módulo multiprocessing posee distintas formas de implementar la ejecución paralela, por ello, para el desarrollo del trabajo se optó por la utilización del objeto Pool del módulo multiprocessing.

Las funciones utilizadas en el trabajo han sido tomadas con el fin de evaluar el tiempo de ejecución de cada una de las instrucciones del código implementado, posteriormente obtener el rendimiento de la ejecución paralela frente a la ejecución lineal.

Para el desarrollo del trabajo se implementaron operaciones matemáticas básicas y funciones matemáticas más complejas como exponenciales logaritmos y funciones trigonométricas; con ello se emplea un algoritmo tipo Benchmarck de evaluación del trabajo y así predecir el tiempo de ejecución del algoritmo.

Debido al corto tiempo de ejecución de las operaciones implementadas, por tanto, fue necesario que el algoritmo realizé un gran número de repeticiones de cada una de las operaciones.

A más de ello se empleó la función “Ventana de Haar” que es comunmente para realizar las pruebas de procesamiento de imágenes con el fin de aumentar la carga computacional, el algoritmo utilizado realiza la detección de rostros y demás caracterirsticas de los rostros, como: ojos, nariz, etc.

1.1 Objetivo general

Evaluar la diferencia entre el procesamiento lineal y paralelo tomando el tiempo de ejecución aplicado a distintas funciones matemáticas.

1.2 Objetivos específicos

1. Comparación entre procesamiento lineal y paralelo en función del tiempo de ejecución implementado con el lenguaje de programación de Python
2. Procesamiento de imágenes mediante el uso y aplicación de distintas funciones implementadas sobre Python.
3. Construcción de un algoritmo (Benchmark) con el fin de evaluar el tiempo promedio de ejecución en función de las operaciones escojidas.

1.3 Alcance

El componente a desarrollar utilizará un conjunto de imágenes de prueba con ciertas características. Un ejemplo de este conjunto es la imagen de Lena Forsén en su tamaño estándar para pruebas de prosesamiento de imágenes de 512x512. Las imágenes de prueba presentan un histograma con un amplio rango dinámico, lo que hace intensivo el

uso de funciones matemáticas, con el fin de generar carga computacional o de procesamiento.

Figura 1 Imagen para prueba "Lena Frosén"

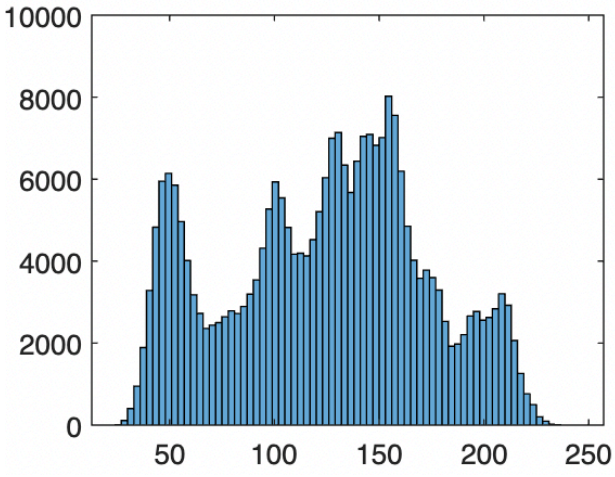


Figura 2 Rango dinámico de la imagen Lena

En la fase de implementación se realizara el preprocesamiento del conjunto de imágenes de prueba, pasando por un proceso de segmentación de una de las imágenes de prueba,

posibilitando de esta forma la implementación del procesamiento paralelo dependiendo del número de segmentaciones realizadas a la imagen de prueba.

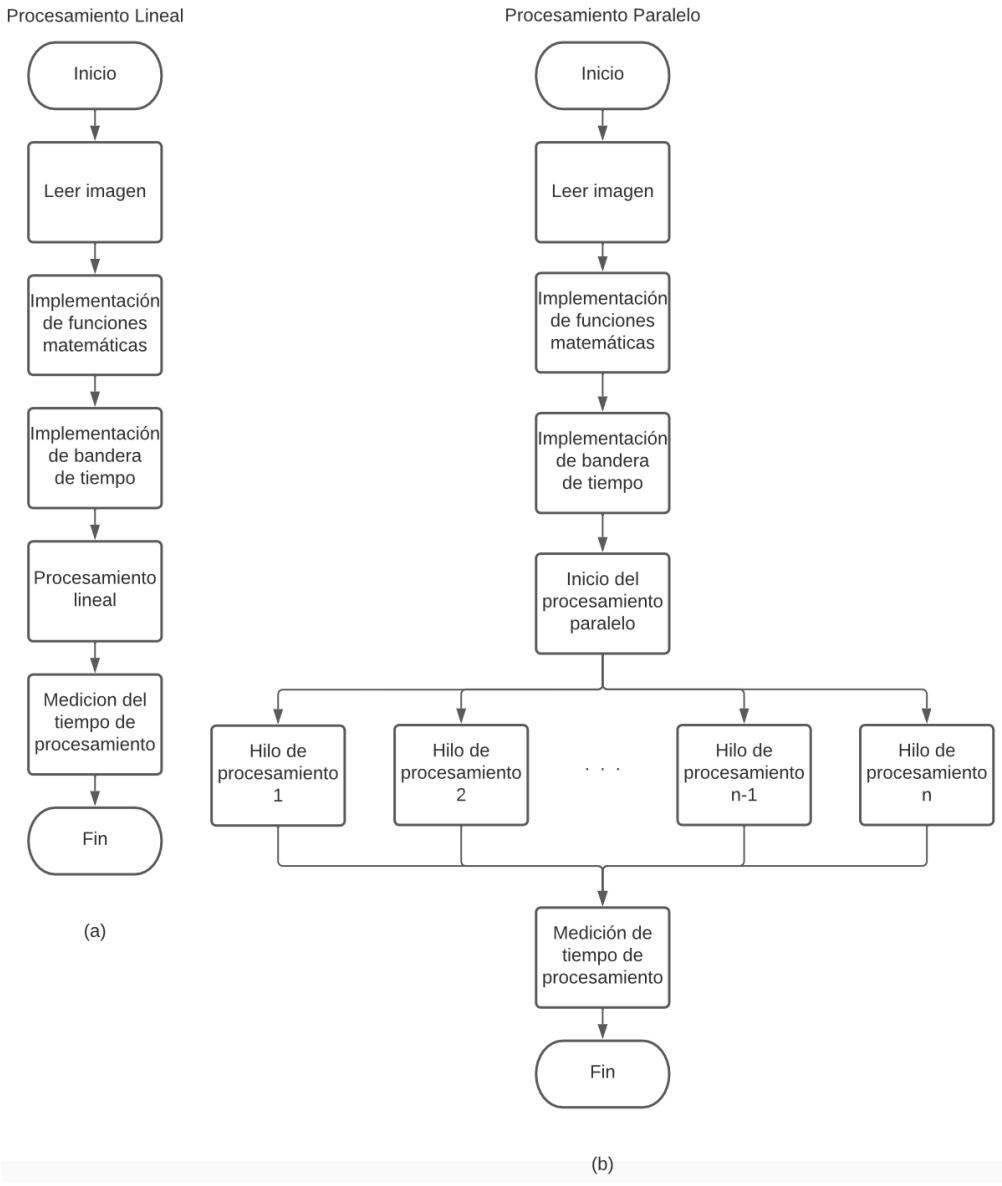


Figura 3 (a) Procesamiento Lineal; (b) Procesamiento Paralelo para n hilos

Los tiempos de ejecución tomados en cada una de las pruebas seran tabulados y comparados para cada una de las funciones implementadas en el trabajo.

1.4 Marco teórico

1.4.1. Python

Python es un software libre y de código abierto, que tiene varias ventajas que lo hacen muy atractivo tanto para uso profesional así como, para aprender a programar.

- Es legible, su sintaxis es elegante y te permite escribir programas mucho más legibles.
- Proporciona un entorno interactivo que facilita las pruebas.
- Su intérprete es gratuito.

Python fue desarrollado por Guidon van Rossum y actualmente está siendo desarrollado continuamente con el aporte de una gran comunidad de colaboradores. En los últimos años, el número de desarrolladores y programadores ha aumentado, lo cual a permitido que Python sea utilizado por un gran numero de empresas, sindo esta una de las razones por las se a conciderado esta aplicación para el desarrollo del presente trabajo.

Multiprocesamiento hace referencia al procesamiento con multiples núcleos, (describir nucleos), cpu-nucleo capaz de realizar todas las operaciones, hoy en dia los procesadores cuentan con esta característica, para hacer refencia al multiprocesamiento en el precente documento se lo describe como Multiprocessing.

Otro mecanismo utilizado en los sistemas de computo modernos se conoce como la Procesamiento basado en hilos siendo esta la ejecución de varios procesos que se ejecutan de manera concurrente bajo un esquema de ejecución por división de tiempo, esto significa que un mismo cpu o núcleo se desarrolla una tarea en intervalos de tiempo siguiendo un hilo de ejecución, en el documento este proceso se lo define como Threading.

En plataformas mas complejas, con procesadores con multipless nucleos, las aplicaciones y principalmente el sistema operativo administran los intervalos de tiempo de ejecucion de los hilos y el uso de los distintos nucleos presentes en el hardware, por lo que no se tiene un control total sobre estas características.

1.4.2. Multiprocesamiento (Multiprocessing)

Multiprocessing es un paquete que admite procesos creados mediante una API similar al módulo Threading. El paquete multiprocessing proporciona simultaneidad, omitiendo efectivamente el Global Interpreter Lock (GIL) mediante el uso de subprocesos en lugar de

hilos de processos. Como resultado, el módulo multiprocessing permite a los programadores aprovechar al máximo las capacidades de varios procesadores (múltiples núcleos) en una máquina determinada. El módulo multiprocessing funciona tanto en sistemas Unix como Windows [1].

El módulo de multiprocessing también tiene APIs sin contraparte en el módulo de threading. Un buen ejemplo de esto es el objeto Pool, que proporciona una manera conveniente de paralelizar la ejecución de funciones para múltiples entradas mediante la distribución de entradas entre procesos (paralelización de datos) [1].

1.4.3. Multi-hilo (Threading)

El procesamiento Multi-hilo ejecuta varios procesos en el tiempo de ejecución, esto se logra debido a la asignación de instantes de tiempo para cada uno de los hilos de ejecución; mientras un proseso se esta ejecutando y se quiere ejecutar otro, se interrumpe el primer procoso dando paso a la nueva petición entrante.

El entorno de Python cuenta con el módulo Threadig que permite aprovechar la ejecución Multi-hilo, este módulo crea interfaces de subprocesos de alto nivel sobre el módulo thread de nivel inferior [2].

1.4.4. Funciones

Como funciones para realizar las pruebas se implementaron las siguientes:

- Suma, Resta, Multiplicación, División, Exponencial, Logaritmo, Seno y Coseno.

```
suma=suma+i
resta=resta-i
multiplicacion=multiplicacion*i
division=10000/(i+1)
exponencial=np.exp(i)
logaritmo=np.log10(i)
seno=math.sin(2*math.pi*i)
coseno=math.cos(2*math.pi*i)
```

Figura 4 Funciones implementadas

2. METODOLOGÍA

Debido al uso multiplataforma de Python, se opta por realizar las pruebas sobre equipos con distintos sistemas operativos, los cuales son: mac OS y Raspberry pi OS.

2.1. Caracteristicas del quipo con mac OS

Para el precente trabajo se hizo uso de un equipo de la marca Apple, el cual cuenta con el procesador M1, que presenta las siguientes características:

- CPU de 8 núcleos con 4 núcleos de rendimiento y 4 de eficiencía
- GPU de 8 núcleos
- RAM de 8 GB

2.2. Caracteristicas del equipo con Raspberry pi OS

Como equipo complementario para el trabajo realizado se ocupo un modelo de la marca HP, especificamente el equipo hp Pavilion de 15”, el cual cuenta con un procesador core i3 de la marca intel, que presenta las siguientes caracteristicas:

- CPU de 2 núcleos
- GPU de 4 núcleos
- RAM de 4 GB

El precente trabajo se enfoca en el desarrollo de un programa sobre el IDLE de la plataforma Python, con el propósito de medir el tiempo de ejecución de ciertos algoritmos implementados. Debido al muy corto tiempo de ejecucuión de las instrucciomes y ciertas operaciones es necesario la implementacion de un programa en el cual se realice la repetición multiples veces, en el presente trabajo se realizaron pruebas entre 1.000.000 y 10.000.000 de operaciones, con el propósito de obtener un tiempo de ejecución, tal que la función utilizada para realizar la medición del tiempo presente resultados suficientemente representativos para una representación gráfica adecuada.

2.3. Librerías

Con el propósito de determinar el tiempo de ejecución de las distintas instrucciones y funciones elegidas, se hizo uso de la librería ”time”, para las operaciones matemáticas se uso las librerías “numpy” y “math” de Python.

```
import numpy as np
import time
import math
```

Figura 5 Librerías utilizadas

El precente trabajo se enfoca en el desarrollo de un programa sobre el IDLE de la plataforma Python, con el propósito de medir el tiempo de ejecución de ciertos algoritmos implementados. Debido al muy corto tiempo de ejecución de las instrucciomes y las grandes capacidades de los equipos desarrollados en la actualidad es necesaria la implementacion de un algoritmo en el cual se realice una repetición multiple de las operaciones implementadas, en el presente trabajo se realizaron pruebas entre 1.000.000 y 10.000.000 de operaciones, con el propósito de obtener un tiempo de ejecución, tal que la función utilizada para realizar la medición del tiempo presente resultados suficientemente representativos para una representación gráfica adecuada.

Con el propósito de determinar el tiempo de ejecución de las distintas instrucciones y funciones elegidas, se hizo uso de la librería ”time”, para las operaciones matemáticas se uso las librerías “numpy” y “math” de Python.

2.4. Funciones implementadas (procesamiento lineal)

La ejecución lineal de la suma se realizó con la implementación del código presente acontinuación:

```
suma=0

for j in range(1000):
    for i in range(10000):
        suma=suma+i
```

Figura 6 Suma (procesamiento lineal)

La ejecución lineal de la resta se realizó con la implementación del código presente acontinuación:

```
resta=10000

for j in range(1000):
    for i in range(10000):
        resta=resta-i
```

Figura 7 Resta (procesamiento lineal)

La ejecución lineal de la multiplicación se realizó con la implementación del código presente acontinuación:

```
multiplicacion=1

for j in range(1000):
    for i in range(10000):
        multiplicacion=multiplicacion*i
```

Figura 8 Multiplicación (procesamiento lineal)

La ejecución lineal de la división se realizó con la implementación del código presente acontinuación:

```
division=0

for j in range(1000):
    for i in range(10000):
        division=10000/(i+1)
```

Figura 9 División (procesamiento lineal)

La ejecución lineal de la exponencial se realizó con la implementación del código presente acontinuación:

```
for j in range(1000):
    for i in range(10000):
        exponencial=np.exp(i)
```

Figura 10 Exponencial (procesamiento lineal)

La ejecución lineal del logaritmo se realizó con la implementación del código presente acontinuación:

```
for j in range(1000):
    for i in range(10000):
        logaritmo=np.log10(i)
```

Figura 11 Logaritmo (procesamiento lineal)

La ejecución lineal del seno se realizó con la implementación del código presente acontinuación:

```
for j in range(1000):
    for i in range(10000):
        seno=math.sin(2*math.pi*i)
```

Figura 12 Seno (procesamiento lineal)

La ejecución lineal del coseno se realizó con la implementación del código presente acontinuación:

```
for j in range(1000):
    for i in range(10000):
        coseno=math.cos(2*math.pi*i)
```

Figura 13 Coseno (procesamiento lineal)

2.5. Módulo implementado (procesamiento paralelo)

La implementación del procesamiento paralelo se hizo uso del módulo multiprocessing con el objeto Pool como se muestra en la figura 11.

```
from multiprocessing import Pool
```

Figura 14 Módulo multiprocessing

Para la utilización del objeto Pool es necesaria la asignación del objeto a una variable y un argumento, el cual hace referencia a la cantidad de grupos para realizar la ejecución paralela; posteriormente se realiza la inicialización de la ejecución paralela con el uso del comando ".map" juntamente con argumentos:

```
p=Pool(5)
p.map(f,[])
```

Figura 15 Asignación para ejecución paralela

Los argumentos del .map:

- f: Función donde se encuentran las operaciones implementadas

- [ ]: Argumentos de la función

La ejecución paralela de las operaciones se implementaron de forma similar a las funciones presentadas en la sección 2.4.

2.6. Tasa de incremento del tiempo de ejcución

Con los datos obtenidos es posible determinar la tasa de incremento del tiempo de ejecución en función del número de operaciones realizadas, para ello se hace uso de la ecuación descrita a continuación.

$$m = \frac{x1 \ - \ x2}{y1 \ - \ y2}$$

Ecuación 2.6.1 Pendiente de una recta

- m: Taza de imcremento
- x1, x2: Puntos del eje x
- y1, y2: Puntos del eje y

2.7. Porcentaje de disminución del tiempo de ejecución

El uso de la ejecución paralela presenta una disminución del tiempo de ejecución de las funciones descritas en la sección 2.4. de este documento.

El cálculo del porcentaje de disminución del tiempo de ejecución se lo realiza con el uso de la ecuación Ec. 2.7.1.

$$\%dism = \frac{Tlin - Tpar}{Tlin} * 100$$

Ecuación 2.7.1. Porcentaje de disminución

- %dism: Porcentaje de disminución del tiempo de ejecución
- Tlin: Tiempo de ejecución lineal
- Tpar: Tiempo de ejecución paralelo

2.8. Intervalos de confianza

Un intervalo de confianza permite calcular dos valores alrededor de una media muestral (uno superior y otro inferior), estos valores acotan un rango de la muestra. [3]
Para el precente trabajo se define un porcentaje de confianza del 95% para el intervalo, para el cálculo del intervalo de confianza se implementa la ecuación 2.8.1.

$$X - \left(\sigma/\sqrt{n}\right) * Z\,\alpha/2 < \mu < X + \left(\sigma/\sqrt{n}\right) * Z\,\alpha/2$$

Ecuación 2.8.1. Intervalo de confianza

- μ: Intervalo de confianza
- X: Media muestral
- σ: Desviación estandar
- n: Tamaño muestral
- $Z\,\alpha/2$: Valor que deja a la derecha un valor de $\alpha/2$

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. Resultados

Los resultados del trabajo se presentan en dos partes; como se dispone a continuación:

- Procesamiento lineal y paralelo (equipo con mac OS)
- Procesamiento lineal y paralelo (equipo con Raspberry pi OS)

3.1.1. Procesamiento lineal y paralelo (equipo con mac OS)

En esta sección se muestran los resultados de la implementación de las distintas funciones descritas en la sección 2.4. implementadas sobre el equipo con el sistema operativo mac OS, los resultados obtenidos se presentan resumidos en la tabla a continuación:

Tabla 1 Tiempos de ejecución en mac OS (Procesamiento lineal)

	# de Operaciones									
	1.000.000	2.000.000	3.000.000	4.000.000	5.000.000	6.000.000	7.000.000	8.000.000	9.000.000	10.000.000
Suma (t[seg])	0,1041	0,1923	0,1923	0,2785	0,2785	0,3523	0,3523	0,447	0,447	0,5396
Resta(t[seg])	0,0918	0,1736	0,246	0,3219	0,3739	0,4491	0,5196	0,5912	0,6587	0,7306
Multiplicación (t[seg])	0,0873	0,1732	0,2369	0,3097	0,3905	0,4538	0,5234	0,592	0,6643	0,7663
División (t[seg])	0,1057	0,2015	0,2874	0,3739	0,4882	0,5581	0,6269	0,7095	0,7975	0,9118
Exponencial (t[seg])	1,019	2,43	3,7985	5,2173	6,5823	7,9756	9,3965	10,7452	12,1459	13,5404
Logaritmo (t[seg])	0,8533	1,6666	2,476	3,2677	4,0967	4,8946	5,6911	6,5038	7,3265	8,0596
Seno (t[seg])	0,2263	0,425	0,6213	0,8004	1,0178	1,2021	1,3824	1,5965	1,7564	1,9447
Coseno (t[seg])	0,2303	0,4214	0,6103	0,8033	0,9987	1,2035	1,41	1,5655	1,7597	1,9437

Para una mejor visualización de los datos obtenidos, se realizarón gráficas con los datos de la “Tabla 1”, los representacione se las dividió en:

Los resultados se agruparon en: operaciones básicas, exponencial y logaritmo, funciones trigonométricas. Esta agrupación permite comparar las plataformas, la ejecución lineal y paralela.

- Operaciones básicas

Figura 16 Operaciones básicas

En la figura 13 se puede observar un comportamiento lineal creciente con respecto al número de repeticiones de las operaciones básicas, donde la tasa de crecimiento

corresponde al tiempo de ejecucion promedio de la funcion analizada, como lo son: suma, resta, multiplicación y división.

- Exponencial y Logaritmo

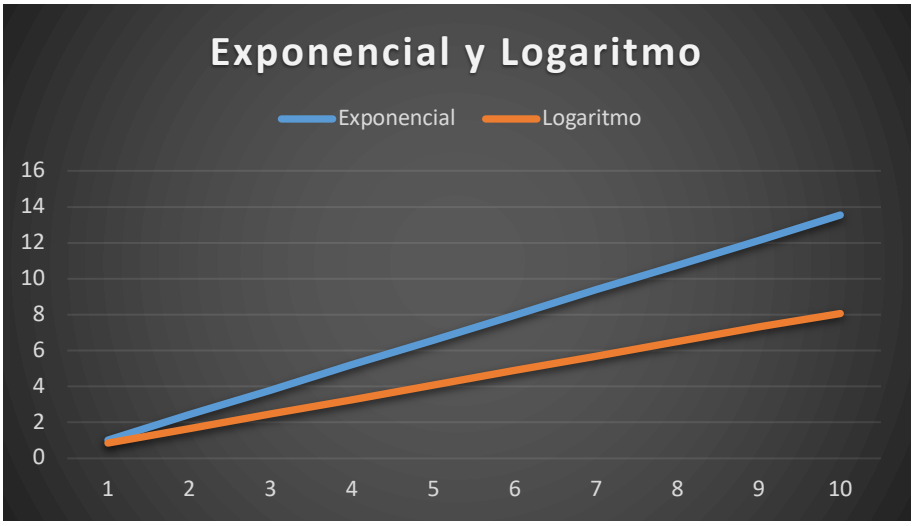


Figura 17 Exponencial y Logaritmo

- Funciones trigonométricas

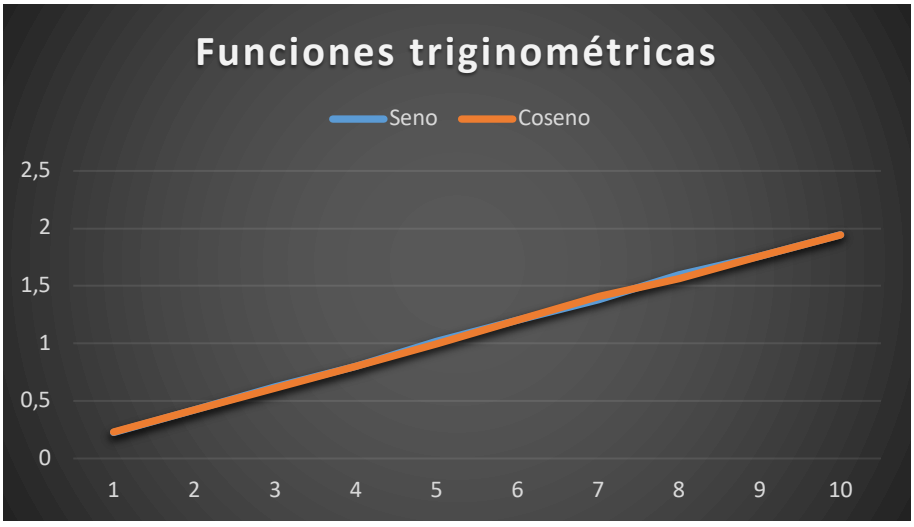


Figura 18 Funciones trigonométricas

3.1.1.1. Intervalos de confianza

Para los datos presentados en la “Tabla 1”, los intervalos de confianza de los datos obtenidos se muestran en la “Tabla 2” presente a continuación.

Tabla 2 Intervalos de confianza (equipo con macOS)

Intervalos de confianza		
Suma (t[seg])	0,8579	0,8819
Resta(t[seg])	0,7152	0,7460
Multiplicación (t[seg])	0,7313	0,8013
División (t[seg])	0,8895	0,9341
Exponencial (t[seg])	13,5284	13,5524
Logaritmo (t[seg])	8,0476	8,0716
Seno (t[seg])	1,9327	1,9567
Coseno (t[seg])	1,9317	1,9557

3.1.1.2. Porcentajes de disminución del tiempo de ejecución

El multiprocesamiento Threading presenta disminución del tiempo de ejecución para un número de hilos pequeños como se observa en las Tablas 3, 5 y 7, mientras mayor es el número de hilos de procesamiento este método de multiprocesamiento es ineficiente.

Por el contrario, el uso de la función Multiprocessing presenta mejoras significantes en cuanto a reducción del tiempo de ejecución de las operaciones matematicas utilizadas, los resultados obtenidos con la ayuda de esta función se muestran en la tabla 4, 6 y 8.

Los resultados presentados en esta sección hacen referencia a las pruebas realizadas sobre el equipo con sistema operativo macOS.

- Suma

Tabla 3 Porcentaje de disminución (Suma-Threading)

# de Hilos	Treading
1	52,62%
2	6,27%
3	-40,06%
4	-86,18%
5	-131,61%

Tabla 4 Porcentaje de disminución (Suma-Multiprocesing)

# de Grupos	Multiprocessing
1	98,87%
2	98,59%
3	98,41%
4	97,82%
5	97,80%

- Exponencial

Tabla 5 Porcentaje de disminución (Exponencial-Threading)

# de Hilos	Treading
1	3,55%
2	-94,02%
3	-182,92%
4	-287,04%
5	-383,76%

Tabla 6 Porcentaje de disminución (Exponencial-Multiprocessing)

# de Grupos	Multiprocessing
1	99,92%
2	99,91%
3	99,89%
4	99,87%
5	99,86%

- Coseno

Tabla 7 Porcentaje de disminución (Coseno-Threading)

# de Hilos	Treading
1	19,75%
2	-59,88%
3	-137,42%
4	-217,74%
5	-305,06%

Tabla 8 Porcentaje de disminución (Coseno-Multiprocessing)

# de Grupos	Multiprocessing
1	99,43%
2	99,34%
3	99,30%
4	99,16%
5	99,11%

3.1.2. Procesamiento lineal y paralelo (equipo con Raspberry pi OS)

En esta sección se muestran los resultados de la implementación de las distintas funciones descritas en la sección 2.4. implementadas sobre el equipo con el sistema operativo Raspberry pi, los resultados obtenidos se presentan resumidos en la tabla a continuación:

Tabla 9 Tiempos de ejecución Raspberry pi OS (Procesamiento lineal)

	# de Operaciones									
	1.000.000	2.000.000	3.000.000	4.000.000	5.000.000	6.000.000	7.000.000	8.000.000	9.000.000	10.000.000
Suma (t[seg])	0,5983	1,3128	1,9807	2,4303	3,1261	4,029	4,408	4,9999	5,7721	6,3498
Resta(t[seg])	0,6386	1,1676	1,952	2,533	3,2394	3,6632	4,3528	4,886	5,593	6,1322
Multiplicación (t[seg])	0,5164	1,0465	1,6019	2,113	2,7349	3,2583	3,6962	4,3233	4,997	5,2545
División (t[seg])	0,1057	0,2015	0,2874	2,6207	3,2427	4,0845	4,6476	5,1614	5,8553	6,4261
Exponencial (t[seg])	1,019	2,43	3,7985	5,2173	6,5823	7,9756	9,3965	10,7452	12,1459	13,5404
Logaritmo (t[seg])	0,8533	1,6666	2,476	3,2677	4,0967	4,8946	5,6911	6,5038	7,3265	8,0596
Seno (t[seg])	0,2263	0,425	0,6213	0,8004	1,0178	1,2021	1,3824	1,5965	1,7564	1,9447
Coseno (t[seg])	0,2303	0,4214	0,6103	0,8033	0,9987	1,2035	1,41	1,5655	1,7597	1,9437

3.1.2.1. Intervalos de confianza

Para los datos presentados en la “Tabla 9”, los intervalos de confianza de los datos obtenidos se muestran en la “Tabla 10” presente a continuación.

Tabla 10 Intervalos de confianza (equipo con macOS)

Intervalos de confianza		
Suma (t[seg])	6,1560	6,5436
Resta(t[seg])	5,7553	6,5091
Multiplicación (t[seg])	5,0793	5,4297
División (t[seg])	6,1985	6,6537
Exponencial (t[seg])	226,4440	226,4680
Logaritmo (t[seg])	86,6700	86,6940
Seno (t[seg])	12,6251	12,6491
Coseno (t[seg])	12,7233	12,7473

3.1.2.2. Porcentajes de disminución del tiempo de ejecución

El multiprocesamiento Threading presenta disminución del tiempo de ejecución para un número de hilos pequeños como se observa en las Tablas 11, 13 y 15, mientras mayor es el número de hilos de procesamiento este método de multiprocesamiento es ineficiente.

Por el contrario, el uso de la función Multiprocessing presenta mejoras significantes en cuanto a reducción del tiempo de ejecución de las operaciones matematicas utilizadas, los resultados obtenidos con la ayuda de esta función se muestran en la tabla 12, 14 y 16.

Los resultados presentados en esta sección hacen referencia a las pruebas realizadas sobre el equipo con sistema operativo macOS.

- Suma

Tabla 11 Porcentaje de disminución (Suma-Threading)

# de Hilos	Treading
1	59,86%
2	8,84%
3	-31,03%
4	-75,70%
5	-120,46%

Tabla 12 Porcentaje de disminución (Suma-Multiprocesing)

# de Grupos	Multiprocessing
1	99,48%
2	99,31%
3	98,99%
4	98,85%
5	98,61%

- Exponencial

Tabla 13 Porcentaje de disminución (Exponencial-Threading)

# de Hilos	Treading
1	-1542,96%
2	-5265,51%
3	-7246,43%
4	-9464,58%
5	-11723,48%

Tabla 14 Porcentaje de disminución (Exponencial-Multiprocessing)

# de Grupos	Multiprocessing
1	99,68%
2	99,60%
3	99,51%
4	99,44%
5	99,40%

- Coseno

Tabla 15 Porcentaje de disminución (Coseno-Threading)

# de Hilos	Treading
1	-308,37%
2	-1073,28%
3	-1516,09%
4	-2150,07%
5	-2749,63%

Tabla 16 Porcentaje de disminución (Coseno-Multiprocessing)

# de Grupos	Multiprocessing
1	99,67%
2	99,63%
3	99,52%
4	99,49%
5	99,39%

3.1.3. Benchmark

El término benchmark es comunmente utilizado en el mundo de la informatica; benchmark hace referencia a la medición del rendimiento de un dispositivo o algún elemento específico del mismo.

En el precente trabajo se ha utilizado una funcion clasica de procesamiento de imágenes debido a que las operaciones matematicasutlilizada por esta funcion han sido caracterizadas y medido su tiempo de ejecución utilizando scripts desarrollados en Python, de esta manera es posible determinar cual sera el tiempo de ejecucion del procesamiento de una imagen en función del tamaño de la matris de la misma para la plataforma computacional utilizada.

Para la utilización de esta funcion de procesamiento de imagen como um Benchmark se tomara como referencia el tiempo de ejecución medido con la plataforma computacional utilizada de esta manera se podra comparar el tiempo de ejecucion en oyras plataformas.

De esta manera se procedio para evaluar el tiempo de ejecución de un computador con las siguientes caracteristicas descritas en la sección 2.2, obteniendose como resultado un tiempo de ejecución que permite establecer la comparación señalada, obteniendose los valores descritos en la tabla 18.

Utilizando Benchmark desarrollado fue posible la comparación de los equipos descritos en las secciones 2.1 y 2.2 obteniendo los siguientes resultados.

Tabla 17 Comparación entre equipos mac OS y Rapsberry pi OS

# de Grupos	Multiprocessing
1	-66,13%
2	-71,35%
3	-78,66%
4	-77,49%
5	-74,73%

La metodologia utilizada ha permitido tener el tiempo de ejecución de varias funciones matemáticas utilizados en diversos algoritmos, por lo que estos datos podrian ser utilizados para construir otras heramientas de evaluación o Benchmarks de diversa indole.

Los datos de tiempo de ejecución obtenidos para las diversas funciones matematicas permiten calcular el tiempo de ejecución referencial de un algoritmo que sera utilizado como Benchmark para la evaluación de otras plataformas computacionales.

3.1.3.1. Ventana de Haar (procesamiento de imágenes)

Ampliamente utilizada en el ámbito del procesamiento de imágenes, para el presente trabajo de lo utilizó para realizar el reconocimiento de rostros, ojos y nariz de la imagen (ver

Figura) “Lena Frosén” que es comunmente utilizada por la característica de amplio rango dinamico con la que cuenta.

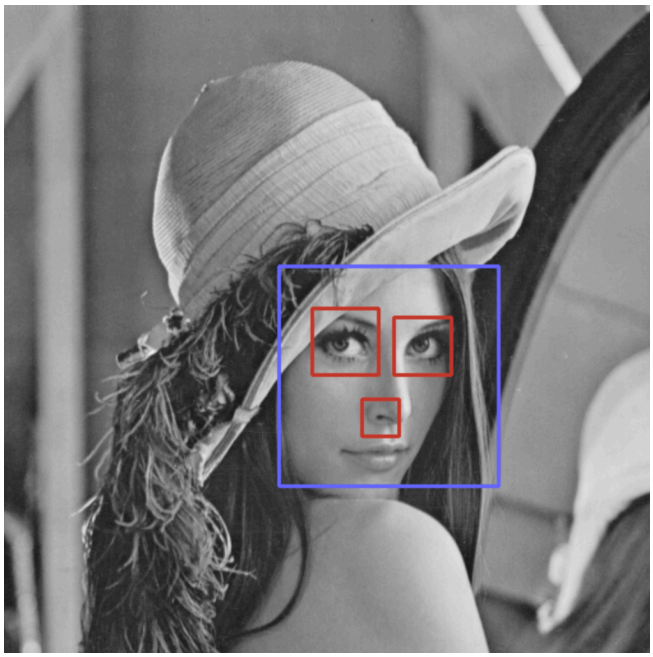


Figura 19 Resultado Ventana de Haar

Para la implementación de la ventana de Haar es necesario el uso de una librería que permita el cambio a escalas de grises de la imágenes que se utilizaran para el reconocimiento de rostros.

```
import cv2
```

Figura 20 Librería cv2

Los tiempo de ejecución obtenidos para el algoritmo con la ventana de Haar se encuentran en la “Tabla 17”, esta tabla contine datos de ejecución lineal, asi como, ejecución paralela (módulo Multiprocessing) y 5 grupos de procesamiento.

Tabla 18 Datos ventana de Haar (a)Lineal; (b)Multiprosessing

(a)

Ventana Haar (t[S])
18,258
18,22
18,184
18,207
18,178

(b)

# de Grupo	Ventana Haar (t[s])
1	0,01128
2	0,01252
3	0,01366
4	0,01639
5	0,02224

3.1.3.2. Construcción de Benchmark

De acuerdo al conocimiento adquido respecto al tiempo de ejecución de las operaciones, es posible determinar el tiempo que tomaría la ejecución de un algoritmo que cuente con ciertas características; con este proposito se empleó un algoritmo el cual dispone un porsentaje variado de las operaciones descritas en el presente trabajo.

La “Tabla 18” contiene datos de la implementación del algoritmo descrito anteriormente, las operaciones escogidas para esto son: Suma, Exponencial y Coseno, para la ejecución del algoritmo se hace uso del módulo multiprocessing y configurado para 5 grupos de procesamiento.

Tabla 19 Datos Benchmarck

Benchmarck (t[s])	Suma	Exponencial	Coseno
0,01968	33%	33%	33%
0,01436	50%	50%	0%
0,01396	50%	0%	50%
0,01391	0%	50%	50%
0,0141	25%	75%	0%
0,01407	75%	25%	0%
0,0146	0%	75%	25%
0,01441	0%	25%	75%

Para la primera implementación se probó una configuración del 33% de cada una de las operaciones y de acuerdo con los datos obtenidos en la sección 3.1.1 fue posible la determinación teórica del tiempo de ejecución del algoritmo implementado.

$$Benchmark(t[s]) = t_{suma} * 0{,}33 + t_{exponencial} * 0{,}33 + t_{coseno} * 0{,}33$$

Ecuación 3.1.3.3.1. Tiempo de ejecución teórico - Benchmark

3.2. Conclusiones

La evaluación del algoritmo para la determinación del tiempo de ejecución de las operaciones fue necesaria, ya que gracias a esto se tomo como medida preventiva el incremento del numero de repeticiones de la operaciones, con el fin de obtener una lectura de datos más apreciables.

La utilizacion del software Python, el cual es de código abierto, es posible la utilización de librerias especificas, para desarrollar segmentos de código para procesamiento lineal, multi-hilo y paralela.

Se pudo observar de la ejecución de los segmentos de código que no se tiene el control de la asignación de los recursos al iniciar el proceso de ejecución tanto lineal, multi-hilo y paralela, sin embargo, se realizaron los desarrollo necesarios para obtener los resultados que permitieron evaluar el tiempo de ejecución de las funciones matemáticas.

El módulo Threading no presentó las caracteristicas requeridas para el trabajo realizado, ya que el tiempo promedio de ejecución con este módulo incrementa en función del incremento del número de hilos de ejecución.

El módulo Multiprocessing de las nuevas versiones de Python presentan una gran mejora al utilizar el objeto Pool, disminuyendo el tiempo de ejecución de las operaciones y funciones descritas en el presente trabajo.

La prueba del algoritmo implementado en una plataforma y equipo de diferentes caracteristicas; el tiempo de ejecución tanto lineal, multi-hilo y paralelo depende del hardware del equipo utilizado.

La construcción de un algoritmo en el cual se pueda implementar varios porcentajes de operaciones y con los datos obtenidos es posible predecir el tiempo de ejecución del nuevo algoritmo.

3.3. Recomendaciones

Implementar el algoritmo desarrollado en eqipos que cuenten con características distintas, con fin de formar un panorama general y asi poder predecir los tiempo de ejecución de cualquiera de las funciones descritas en el presente trabajo.

El módulo Threading posee ciertas restricciones respecto a la implementación de ejecución multi-hilo, por el el uso de este módulo para la realización de trabajor futuros no es recomendable.

Hacer uso de las distintas configuraciones del módulo Multiprocessing, ya que es posible alcanzar una disminución del tiempo de ejecución paralela.

Generar mayor carga computacional con el fin de comprobar un comportamiento similar o no que se ha observado en trabajo desarrollado.

REFERENCIAS BIBLIOGRÁFICAS

[1] Python, «Multiprocessing,» 22 Noviembre 2022. [En línea]. Available: https://docs.python.org/3/library/multiprocessing.html. [Último acceso: 26 Noviembre 2022].

[2] Python, «Threading,» 22 Noviembre 2022. [En línea]. Available: https://docs.python.org/3/library/threading.html. [Último acceso: 26 Noviembre 2022].

[3] F. J. M. Sanjuán, «economipedia,» 02 10 2017. [En línea]. Available: https://economipedia.com/definiciones/intervalo-de-confianza.html. [Último acceso: 14 12 2022].

[4] Python, «Python,» 2022. [En línea]. Available: https://www.python.org/. [Último acceso: 26 Noviembre 2022].