TESIS

UNIVERSIDAD CIENTÍFICA DEL SUR

UNIVERSIDAD CIENTÍFICA DEL SUR

FACULTAD DE CIENCIAS EMPRESARIALES

CARRERA PROFESIONAL DE

INGENIERÍA DE SISTEMAS DE INFORMACIÓN Y GESTIÓN

"DESARROLLO DE LIBRERÍA PARA PROCESAMIENTO Y GESTIÓN DE IMÁGENES USANDO OPENMPI Y OPENMP, LIMA 2023"

Tesis para optar el título profesional de:

INGENIERO DE SISTEMAS DE INFORMACIÓN

Presentado por:

Robinson Oliva Salazar (0000-0002-1376-5112)

Asesor:

Wilver Auccahuasi Aiquipa (0000-0001-8820-4013)

Lima - Perú

2023

Tesis publicada con autorización del autor
No olvide citar esta tesis

ACTA DE SUSTENTACIÓN DE TESIS

Siendo las 09:15 del día 28 de junio del 2024, en la sala de conferencias se reunieron los miembros del Jurado:

Los integrantes del Jurado de tesis:

Presidente	**Carlos Enrique López Sovero**
Miembro 1	**Robert Roy Saavedra Jiménez**
Miembro 2	**Luis Enrique Acosta Medina**

Se reúnen para evaluar la tesis titulada:
"DESARROLLO DE LIBRERÍA PARA PROCESAMIENTO Y GESTIÓN DE IMÁGENES USANDO OPENMPI Y OPENMP, LIMA 2023"

Presentado por el bachiller:
Robinson Oliva Salazar

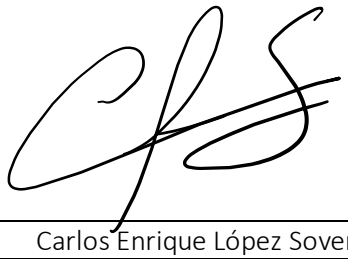
Para optar al título profesional de
Ingeniería de Sistemas de Información y Gestión

Asesorados por:
Wilver Auccahuasi Aiquipa

Luego de haber evaluado el informe final de tesis y evaluado el desempeño del estudiante de la Carrera de **Ingeniería de Sistemas de Información y Gestión** en la sustentación, concluyen de manera unánime ( ) por mayoría simple ( ) calificar a:

Tesistas:	**Robinson Oliva Salazar**		
Nota (en letras):	**Catorce**		
Aprobado (X)	Aprobado - Muy buena ()	Aprobado - Sobresaliente ( )	Desaprobado ( )

Los miembros del jurado firman en señal de conformidad.

	
Carlos Enrique López Sovero	Wilver Auccahuasi Aiquipa
Presidente del Jurado	*Asesor*
	
Robert Roy Saavedra Jiménez	Luis Enrique Acosta Medina
Miembro 1	*Miembro 2*

ANEXO: FORMATO DE DECLARACIÓN DE ORIGINALIDAD DE CONTENIDO DE INFORME FINAL DE TESIS O TRABAJO DE INVESTIGACIÓN

UNIVERSIDAD CIENTÍFICA DEL SUR

DECLARACIÓN DE ORIGINALIDAD DE CONTENIDO DE INFORME FINAL DE TESIS / TRABAJO DE INVESTIGACIÓN[1]

Lima, 19 de Julio del 2024

Señor,
Dr. Carlos Zavalaga Reyes

Director General de Investigación, Desarrollo e Innovación
Universidad Científica del Sur
Presente. –

De nuestra consideración,

Yo: Robinson Oliva Salazar, estudiante/egresado de la carrera de Ingeniería de Sistemas de Información y Gestión de la Universidad Científica del Sur, en conjunto con el asesor/la asesora de tesis Ing. Wilver Auccahuasi Aiquipa, declaramos que este informe final de tesis/trabajo de investigación titulado: "Desarrollo de Librería para Procesamiento de Imágenes usando OpenMPI y OpenMP", sustentado para obtener el título/grado de Ingeniero de Sistemas de Información es original.

Es decir, no contiene plagio parcial ni total, cuando se utilizó información de fuentes externas se reconoció la autoría mediante la adecuada citación y los resultados obtenidos son producto entero de mi (nuestra) investigación y no han sido falseados ni fabricados. Todo esto en cumplimiento del Código de ética en la investigación, Reglamento del Comité de Integridad Científica, Reglamento de Propiedad Intelectual, Normas y procedimientos de los trabajos de investigación para la obtención de títulos profesionales y grados académicos [2], que afirmamos conocer en su totalidad.

Por ello, en caso de identificarse alguna situación de plagio, falsificación o fabricación de resultados, nos allanamos al proceso de investigación que establezca la Universidad y las posibles sanciones que pudieran surgir.

Firmamos en conformidad con lo declarado,



Firma tesista
DNI: 09418075



Firma del asesor de tesis
DNI: 43375865

[1] En conformidad con la Resolución del Consejo Directivo N° 084-2022-SUNEDU/CD
[2] Los documentos están en: https://investigacion.cientifica.edu.pe/reglamentos/

INFORME DE REVISIÓN DE ORIGINALIDAD

Título del documento evaluado. Desarrollo de librería para procesamiento y gestión de imágenes usando OpenMPI y OpenMP, Lima 2023

Autores. Robinson Oliva Salazar

Mecanismo de revisión de originalidad. Evaluación con Turnitin (ID 2445426751).

Resultado de la evaluación. 4%

Revisado por. Magaly Kelly Guerrero Huaracallo.

Comentarios sobre la revisión. Filtros usados: excluir fuentes de menos de 12 palabras.



TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Contenido

1.0 INTRODUCCION **1**

2.0 REALIDAD PROBLEMÁTICA **6**

3.0 FUNDAMENTO TEORICO **7**

4.0 JUSTIFICACION DE LA INVESTIGACION **15**

5.0 OBJETIVOS DE LA INVESTIGACION **16**

6.0 HIPOTESIS DE LA INVESTIGACION **16**

7.0 METODOS Y MATERIALES **17**

8.0 RESULTADOS **21**

9.0 DISCUSION **29**

10.0 CONCLUSIONES **31**

11.0 REFERENCIAS **32**

12.0 ANEXOS **34**

Anexo 1: Matriz de consistencia **35**

Anexo 2: Matriz de operacionalización **36**

Anexo 3: Ficha de observación/instrumento de evaluación **37**

Anexo 4: Programa haciendo uso de la librería hibrida **40**

Anexo 5: Sistema de Información y Datos del Sistema de Observación Terrestre **41**

Anexo 6: Evidencias de envío de articulo a congreso para su publicación en revista científica **42**

Anexo 7: Evidencias de aceptación de articulo en congreso para su publicación en revista científica **43**

Anexo 8: Resolución Directoral de Carrera **44**

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

ÍNDICE DE TABLAS

Tabla 1-Datos estadísticos para el cálculo t student con el tiempo de ejecución 26

Tabla 2-Datos estadísticos para el cálculo t student con el tamaño de archivo 28

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

ÍNDICE DE FIGURAS

Figura 1-Ejemplo de programa Hola mundo MPI 10

Figura 2-Ejemplo de programa Hola mundo OpenMP 12

Figura 3-Ejemplo de árbol binario 15

Figura 4-Fases de la metodología 19

Figura 5-Esquema de funcionalidades de la librería 20

Figura 6-Fragmento en Python para el cálculo p-value respecto a al tiempo de ejecución Shapiro-Wilk 22

Figura 7-Fragmento en Python para el cálculo p-value respecto al tamaño de archivo Shapiro-Wilk 22

Figura 8-Fragmento en Python para visualizar el grafico Q-Q respecto al tiempo de ejecución 23

Figura 9-Grafico Q-Q para las diferencias entre preprueba y posprueba en el caso de la medida del tiempo de ejecución 23

Figura 10-Fragmento en Python para visualizar el grafico Q-Q respecto al tamaño de archivo 24

Figura 11-Grafico Q-Q para las diferencias entre preprueba y posprueba en el caso de la medida del tamaño de archivo 24

Figura 12-Grafico tiempos de ejecución en preprueba 25

Figura 13-Grafico tiempos de ejecución en posprueba 26

Figura 14-Grafico tamaño de archivo comprimido en preprueba 27

Figura 15-Grafico tamaño de archivo comprimido en posprueba 28

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

TÍTULO DEL PROYECTO (HOJA SEPARADA)

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

DESARROLLO DE LIBRERÍA PARA PROCESAMIENTO DE IMÁGENES USANDO OPENMPI Y OPENMP

Robinson Oliva Salazar[1] and Wilver Auccahuasi[2]

[1]Ingeniería de Sistemas de Información y Gestión, 100040315@cientifica.edu.pe Facultad de Ciencias Empresariales, Universidad Científica del Sur, Lima, Perú.
[2]Docente investigador asociado, wauccahuasi@cientifica.edu.pe, Facultad de Ciencias Empresariales, Universidad Científica del Sur, Lima, Perú.

RESUMEN

Hoy en día es común encontrar procesadores con más de un núcleo en cualquier dispositivo. Este avance es muy importante si se desea implementar soluciones para dispositivos embebidos donde los recursos de sistema son menores. Dado la presencia de más un núcleo en los modernos procesadores, resulta propicio aprovechar estas características bajo el enfoque de la programación paralela. El estudio se enfoca en la implementación de una librería hibrida usando las tecnologías OpenMPI y OpenMP con el objetivo de reducir el tiempo de compresión de imágenes bajo el algoritmo de Huffman, en el cual no se tiene perdidas de información. Para la investigación se empleó un enfoque cuantitativo ya que es necesario realizar mediciones a la experiencia, con un diseño de tipo experimental siendo el objetivo general, determinar la influencia de la librería hibrida sobre el procesamiento de imágenes al comprimir la información. La implementación de la librería hibrida está basada en los enfoques del paralelismo de datos y paralelismo por tareas, complementándose ambas de manera coordinada. Se ha tomado mediciones antes y después de la aplicación de la librería hibrida correspondiendo a la clase preexperimental del diseño experimental. La evaluación de los casos preprueba y posprueba se realizó mediante las fichas de observación. Se utilizó una muestra de 30 imágenes satelitales obtenidas del Sistema de Información y Datos del Sistema de Observación Terrestre (EOSDIS en inglés) las cuales se referencian en el anexo 5. Los resultados mostraron un mejor desempeño en cuanto al tiempo de ejecución para la compresión de la imagen cuando se emplea la librería hibrida. Se observo un menor tiempo de ejecución, pero no existiendo mejoras significativas en cuanto a la eficiencia en el tamaño final del archivo comprimido. El análisis estadístico, incluyéndose las pruebas de hipótesis, confirman la significancia de la mejora en la dimensión del tiempo de ejecución. En conclusión, el desarrollo de la librería hibrida ha demostrado ser eficaz para reducir el tiempo de ejecución de la compresión de imágenes mediante el algoritmo de Huffman. A pesar de no existir mejoras significativas en el ratio respecto a los tamaños del archivo antes y después de la comprensión de la imagen.

Palabras clave: Tecnologías de Información y Comunicación, Algoritmo de Huffman, OpenMPI, OpenMP, Procesamiento paralelo, compresión de imágenes

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

ABSTRACT

Nowadays it is common to find processors with more than one core in any device. This advance is very important if you want to implement solutions for embedded devices where system resources are less. Given the presence of more than one core in modern processors, it is advisable to take advantage of these characteristics under the approach of parallel programming. The study focuses on the implementation of a hybrid library using OpenMPI and OpenMP technologies with the objective of reducing image compression time under the Huffman algorithm, in which there is no information loss. For the research, a quantitative approach was used since it is necessary to make measurements of the experience, with an experimental design, the general objective being to determine the influence of the hybrid library on image processing by compressing the information. The implementation of the hybrid library is based on the approaches of data parallelism and task parallelism, complementing each other in a coordinated manner. Measurements have been taken before and after the application of the hybrid library corresponding to the pre-experimental class of the experimental design. The evaluation of the pre-test and post-test cases was carried out using observation sheets. A sample of 30 satellite images obtained from the Earth Observation System Data and Information System (EOSDIS) was used which are referenced in annex 5. The results showed better performance in terms of execution time for image compression when the hybrid library is used. A shorter execution time is observed, but there are no significant improvements in efficiency in the final size of the compressed file. Statistical analysis, including hypothesis testing, confirms the importance of the improvement in the execution time dimension. In conclusion, the development of the hybrid library has proven to be effective in reducing the execution time of image compression using the Huffman algorithm. Despite there being no significant improvements in the ratio of file sizes before and after image comprehension.

Keywords: Information and Communication Technologies, Huffman algorithm, OpenMPI, OpenMP, Parallel processing, image compression

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

1.0 INTRODUCCION

Actualmente los fabricantes de microprocesadores ya no están preocupados por conseguir mayor frecuencia en sus dispositivos, siendo la tendencia actual el de fabricar microprocesadores con varios núcleos. El diseño de más de un núcleo ayuda a mejorar el desempeño si se considera que el procesamiento se realiza de manera paralela. En ese contexto, es de gran importancia que los programas también tengan ese esquema de arquitectura paralela para poder aprovechar estas ventajas tecnológicas.

Las implementaciones de software relacionadas a procesamiento de imágenes de gran tamaño generalmente son desarrolladas en lenguaje C o C++ haciendo referencia a librerías OpenMP ó MPI. En ello, un aspecto importante a considerar son los algoritmos de compresión que se utilizan en el tratamiento de imágenes y la técnica de paralelización que se propone va estar estrechamente relacionada con el tipo de algoritmo empleado.

Al referirnos a los algoritmos, resulta interesante aquellos en los que se tiene menor perdida de información en las trasformaciones internas manteniendo así la calidad de la información. Uno de los algoritmos que resulta interesante por cumplir con las características mencionadas anteriormente es el algoritmo de Huffman y también por la característica de no presentar complejidad en su implementación resultan apropiados para ser implementados en los sistemas embebidos donde los recursos del sistema son limitados. Por ejemplo, en la investigación realizada por Zhao, Y (2022) se muestra la versatilidad que tiene el algoritmo de Huffman al momento de la compresión de la información en sistemas embebidos. En su investigación, los datos privados son encaminados hacia un codificador de Huffman diseñado para comprimir los datos, se escogió esta compresión debido a que no existe perdidas al recuperar la información. Se usaron 200 muestras aleatorias de texto que fueron generadas por la librería Lorem-Ipsum. De ellos, 100 textos tuvieron un tamaño de 5000 bytes y los otros 100 tuvieron un tamaño de 10000 bytes. Con este trabajo se evidencia el desempeño de la codificación Huffman en un ambiente reducido de recursos como lo es una memoria flash.

Sin embargo, el algoritmo de Huffman es aún ineficiente cuando se manejan volúmenes grandes de información en los sistemas embebidos pese a que es considerado como uno en los que no se considera perdidas en el proceso de compresión tal como lo manifiesta Li et al. (2020). Por ejemplo, en su investigación acerca de los datos magnéticos en el espacio, en lugar de usar algoritmos de compresión basados en diccionario o entropía los cuales tratan los datos como cadena de texto se propone una codificación diferencial ya que hay datos que son numéricos combinando así con una codificación canónica de Huffman. En esta codificación, los caracteres con alta frecuencia serán representados por bit de longitud menor. A pesar que puede reducir el tamaño de la data, su tabla de codificación que mapea los caracteres a codificar puede ocupar un tamaño considerable por lo que puede impactar en la eficiencia para el tiempo de ejecución del algoritmo. En ese sentido, una manera de optimizar el tiempo de procesamiento podría ser utilizando las técnicas de paralelización.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Los algoritmos aritméticos, el algoritmo de Lempel-Ziv 77 (LZ77), la codificación por Huffman y los algoritmos de precisión (K-RLE) son algoritmos de compresión de datos sin pérdidas pero que tienen una tasa de rendimiento baja debido a su complejidad en el procesamiento y también al tiempo de ejecución. Tal es así que Mahammad & Viswanatham (2020) realizaron una investigación presentando una aproximación al esquema del procesamiento paralelo con el propósito de reducir el tiempo de ejecución para los algoritmos de compresión antes mencionados. Propusieron un modelo de paralelismo basado en OpenMP para los algoritmos de compresión. Dicho experimento fue realizado usando las librerías Intel Parallel studio en plataforma Core 2 duo—2.4 GHz, 1 Gb RAM. Se analizaron con archivos cuyos tamaños eran: 1000 Kb, 3000 Kb, 5000 Kb, 10000 Kb y 15000 Kb. Se llego a la conclusión que, a pesar de ser algoritmos pesados, el procesamiento en paralelo usando OpenMP es una técnica prometedora que mejora la performance del proceso de compresión.

El incremento en la proporción del tráfico de video en internet ha incentivado los esfuerzos por mejorar los estándares de compresión de video para minimizar el ancho de banda mientras que se mantienen los niveles altos en la calidad. De ahí que Migallón et al (2021) demostraron que los algoritmos paralelos son particularmente adecuados para plataformas de memoria compartida, y puede reducir significativamente su enorme complejidad computacional. En su investigación se usó el compilador GCC y OpenMP. Los algoritmos fueron probados en cuatro secuencias de video cada una de las cuales con diferente resolución. En todo el experimento se usaron el mismo número de frames para codificar (120) para todas las secuencias de video. Adicionalmente a la eficiencia del procesamiento en paralelo se demostró que un correcto balanceo de la data al ser codificada lleva a una significante mejora en la escalabilidad.

Existe una gran demanda por la compresión de datos debido a la necesidad de disminuir el tiempo de transmisión e incrementar la capacidad de almacenamiento de datos. La compresión de datos es una técnica para representar información tal como imágenes, videos y otros en formato compacto. Hay una gran variedad de técnicas que permiten mantener la información lo más cercana a su forma original usando una menor cantidad de bits para enviarlos por un canal de comunicación. Zhao et. al (2020) señala que las técnicas del paralelismo por lo general son empleadas para resolver problemas complejos de la realidad como por ejemplo la paralelización de programas como STRUCTURE y NEWHYBRIDS, los cuales son usados en temas de estructura genética. En este caso se abordó el problema procesando tareas en paralelo en un procesador multihilo.

Según Shaharim et al (2022), la compresión de imágenes satelitales basadas en el algoritmo estándar CCSDS-MHC con soporte a procesamiento paralelo ha demostrado ser muy eficiente en tiempo de ejecución. Para la paralelización del algoritmo estándar CCSDS-MHC se trató de encontrar las correctas secciones del algoritmo que se podrían paralelizar. A diferencia de trabajos previos orientados a desktop y GPU, en este trabajo se enfocaron en las características multicore de los procesadores que podrían usarse en las plataformas de satélites. El primer paso fue convertir a C/C++ el algoritmo CCSDS-MHC que inicialmente esta por estándar en java, luego el algoritmo es paralelizado usan OpenMP. Se obtuvo que la compresión en imágenes pequeñas era poco, esto lo atribuyen a que el tamaño de la palabra clave era cercanamente igual para dichas imágenes.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Alluru & Jeganathan (2021) comentan que a través de los años se han realizado muchos esfuerzos por mejorar el rendimiento en los mecanismos de paralelización, para ello se ha diseñado compiladores que permitan la auto paralelización. Manifiestan que en un compilador paralelizante, las transformaciones de código ayudan a reducir las dependencias de datos y a identificar el paralelismo en el código. En su investigación, los autores emplean un modelo llamado identificador de dependencia de datos en el cual un programa es representado por un grafo y mediante ello poder encontrar sus datos dependientes. Usando este modelo los autores consiguen otros algoritmos para movimiento de código invariable en bucle, análisis de rango en vivo, división de nodos y transformaciones de código de fusión de bucle. Todas las optimizaciones que se discutieron se justifican como validado conceptualmente con una secuencia de rigurosos teoremas. Estas demostraciones teóricas sirven al propósito de realizar fácilmente las optimizaciones de un programa.

Süß et al (2021) señala que aún existen deficiencias en los métodos de auto paralelización ya que los bucles deben ser analizados estáticamente y muchas veces se prohíbe las llamadas a funciones dentro de los bucles. Así introdujeron una extensión del lenguaje C el cual permite paralelizar cualquier bucle conteniendo llamada a funciones. Por otro lado; Mustafa (2022) señala que la combinación de un auto paralelizador y un sistema de tuneo garantiza una buena performance; sin embargo, elegir una herramienta de tuneo para usar depende del propósito del estudio/investigación y de las características de la herramienta.

Aldinucci et. al (2021) propone una metodología que consta de cuatro pasos para paralelizar algunas aplicaciones del ámbito científico: 1) Identificar todos los bucles paralelizables en el código con una buena estrategia de búsqueda; 2) Evaluar la ganancia potencial del rendimiento obtenible al modificar cada bucle paralelizable, filtrando aquellos que no vale la pena paralelizar; 3) Hacer que cada uno de los bucles candidatos restantes sea autocontenido para eliminar las verdaderas dependencias de datos entre las diferentes iteraciones; 4) Utilizar bucles no paralelizables para implementar una lógica de checkpointing. Con estas recomendaciones demostraron que la misma metodología funciona para la paralelización en el modelo de memoria compartida (mediante OpenMP), el modelo de paso de mensajes (mediante MPI) y el modelo de computación de propósito general en GPU (mediante OpenACC).

Harel, Pinter & Oren (2022) indican que OpenMP es la técnica más usada debido a su interfaz legible. Sin embargo, la introducción de OpenMP en el código, especialmente el código legacy, es un desafío debido a los obstáculos en la gestión de la memoria compartida. Gonnord et. al (2022) explican que el principal problema en la gestión de memoria es el no determinismo, el cual se produce cuando dos a más hilos en un único proceso acceden a la misma dirección de memoria concurrentemente.

Un nuevo modelo de paralelismo es el propuesto por Yin & Shi (2022) para el caso de procesamiento de gran volumen de datos heterogéneos basados en una combinación de MPI y OpenMP. En su investigación proponen la combinación de MPI y OpenMP de manera que el modelo hibrido analiza la información, divide las tareas en paralelo y bloquea el cache paralelo. Adicionalmente en su investigación establecieron un mecanismo de balanceo de carga paralela para dividir de manera óptima la carga heterogénea, a la que combinada con el modelo hibrido MPI/OpenMP se consigue

Tesis publicada con autorización del autor
No olvide citar esta tesis

optimizar la comunicación paralela. Para conseguir el paralelismo, el hilo principal con el cual comienza el programa es serial y múltiples hilos comienzan a trabajar juntos. Con MPI en la capa alta y OpenMP en la capa baja; el MPI primero descompone el problema de computación paralela en múltiples tareas y asigna a diferentes nodos subrutinas. Luego las instrucciones OpenMP embebidas en los nodos llevan a cabo la distribución paralela de la ejecución. El experimento fue realizado en un nodo compuesto por 2 CPUs y un GPU los cuales son conectados a través de un bus MPI y por el bus PCI. Como datos se usó el conjunto heterogéneo Cityscapes (en total se empleó 5000 datos heterogéneos), el sistema operativo usado fue Linux y el compilador con el que se construyó el programa fue gcc. Los datos heterogéneos procesados con algoritmos de paralelismo presentaron buena eficiencia.

He et. al (2022) comentan que, para conseguir buenos resultados con la aplicación del paralelismo, hay que tener un buen conocimiento del fenómeno o conceptualización de lo que se quiere paralelizar. Tal es así, que lo aplicaron al estudio sobre la inestabilidad de decaimiento paramétrico (PDI por sus siglas en inglés), el cual es un importante fenómeno físico en el calentamiento ionosférico. La simulación del fenómeno requiere de mucho tiempo de ejecución debido a que es una simulación numérica. Los autores también propusieron una interfaz híbrida MPI/OpenMP, con este esquema pudo demostrar su flexibilidad y portabilidad. Además, el tiempo de simulación pudo reducirse considerablemente. Todavía una mejora al esquema propuesto fue dada por Rosenberg et. al (2020) en la que indica una aplicación del esquema CUDA al esquema hibrido MPI/OpenMP proporciona una mejora significativa frente a los cálculos realizados en la CPU.

Investigaciones mencionadas anteriormente como las realizadas por Li et al. (2020) muestran, que el algoritmo de Huffman resulta muy apropiado para la compresión de datos sin pérdidas frente a otros algoritmos que son difíciles de adaptar a la plataforma embebida. Las investigaciones de Abbasi et al. (2023) y Li et al. (2020) aun muestran que, para grandes volúmenes de información, los algoritmos de compresión aún no están optimizados. Al ser un algoritmo de dos pasadas: la primera para encontrar coincidencias estadísticas y la segunda para la codificación, esto trae implicancias en el tiempo de ejecución. Para lo cual, en el presente proyecto se propone el empleo de la programación paralela aprovechando la tecnología de multinúcleo en los procesadores actuales.

Como se mencionó anteriormente, Hadjidoukas et. al (2020) manifiestan que el paralelismo basado en tareas se reconoce como una de las principales formas de paralelizar un código, en este enfoque las tareas asíncronas son lanzadas y distribuidas a través de unidades de procesamiento de una maquina local, un clúster o una supercomputadora. Las tareas pueden estar completamente desacopladas, es decir corresponder a un conjunto de tareas independientes, o de lo contrario ser parte de un algoritmo iterativo donde las tareas resultantes son procesadas y conducidas a un siguiente paso. Los autores, en su trabajo de investigación introdujeron torcpy, una librería con plataforma agnóstica adaptativa al balanceo de carga la cual orquesta la ejecución de las tareas de manera asíncrona sobre plataformas con memoria compartida y distribuida. La librería fue implementada sobre MPI y multihilo. Para el esquema de

Tesis publicada con autorización del autor
No olvide citar esta tesis

comunicación distribuida emplearon la técnica del maestro orquestador y los hilos trabajadores los cuales extraen las tareas desde una implementación con colas. Esta conceptualización puede resultar compleja si se aplica únicamente a imágenes sin un entorno de cluster.

Los antecedentes mencionados muestran la importancia de paralelizar los procedimientos y los diversos esfuerzos tanto a nivel de tecnología como en algoritmos. Muchas de las investigaciones mencionadas han utilizado las interfaces OpenMPI y MPI en los cuales se trabaja independientemente y en algunos casos de manera hibrida (Hadjidoukas et. al, 2020). Pero las investigaciones están enfocadas a sistemas con grandes recursos como en el caso de clúster de computadoras, sin embargo, estos recursos no son disponibles por ejemplo en los sistemas embebidos donde es necesario optimizar los procesos y las herramientas de compilación. El empleo del algoritmo de Huffman aun muestra versatilidad por su sencillez al implementarlo pese a su antigüedad y se podría combinar con las técnicas OpenMPI y OpenMP aplicado a este algoritmo con el propósito de mejorar el desempeño y conseguir un mejor desempeño en el procesamiento de imágenes. En la combinación de tecnologías MPI y OpenMP, con el MPI se podría aprovechar de manera directa cada uno de los núcleos que tiene un moderno microprocesador multinúcleo a diferencia de la tecnología OpenMP en donde se emplea la memoria compartida.

En el ámbito local, Piñán (2018) demuestra que el uso de la computación paralela puede disminuir el tiempo de ejecución de los programas. En su trabajo, implemento un clúster para mostrar que un determinado programa se puede ejecutar en dos nodos bajo el sistema operativo Linux. Para ello utilizo Pelican HPC Linux en máquinas virtuales con los que creo dos nodos. Sin embargo, la implementación de un sistema como clúster es complejo si queremos aplicarlo a sistemas embebidos como en el caso de satélites donde los recursos del propio sistema son limitados.

Por otro lado, Gómez (2020) realizo un comparativo entre las tecnologías MPI vs OpenMP aplicando a resolver matemáticamente el conjunto de Mandelbrot. El estudio fue aplicado sobre las ecuaciones recursivas del problema matemático; se utilizó una HP Notebook con procesador AMD Ryzen 5 Quad-Core y memoria de 8GB. Se construyo con el compilador GNU C/C++ sobre el sistema operativo linux OpenSUSE de 64 bit. Las pruebas se ejecutaron con diferentes límites de iteración y numero de procesadores. Se mostro que las implementaciones con OpenMP tuvieron mejor rendimiento que las implementaciones en MPI. En este ámbito se puede comentar que las tecnologías tanto OpenMP y MPI han sido comparadas independientemente, pero trabajando en conjunto ambas tecnologías repotencian los beneficios de la programación paralela.

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

2.0 REALIDAD PROBLEMÁTICA

La compresión de archivos de imagen es crucial en dispositivos embebidos por varias razones:

a) Limitaciones de almacenamiento:

Los dispositivos embebidos, como sensores, cámaras o microcontroladores, a menudo tienen restricciones de espacio de almacenamiento. Comprimir imágenes reduce la cantidad de memoria necesaria para almacenarlas, permitiendo que los dispositivos funcionen eficientemente con recursos limitados.

b) Ancho de banda y transmisión:

En aplicaciones donde se transmiten imágenes (por ejemplo, cámaras de seguridad o sistemas de monitoreo), la compresión reduce el ancho de banda necesario. Menor cantidad de datos transmitidos significan una comunicación más rápida y eficiente.

c) Eficiencia energética:

La compresión disminuye la cantidad de datos que deben procesarse y transferirse. Esto reduce el consumo de energía en dispositivos con batería, como cámaras de teléfonos móviles o sistemas de vigilancia inalámbricos.

d) Tiempo de procesamiento:

Al comprimir imágenes antes de procesarlas, se acelera el tiempo de cálculo. Menos datos implican menos operaciones y menos tiempo de procesamiento.

El algoritmo de Huffman es un algoritmo en donde no se considera perdidas en el proceso de compresión. Sin embargo, el algoritmo de Huffman es aún ineficiente cuando se manejan volúmenes grandes de información en los sistemas embebidos por lo que puede impactar en la eficiencia para el tiempo de ejecución al ser un algoritmo basado en estadística tal como lo manifiesta Li et al. (2020)

Otra problemática hace referencia en los sistemas embebidos relacionados con el internet de las cosas, por ejemplo, en el procesamiento de imágenes cada vez más grandes y bajo algoritmos sin pérdida de información como lo es Huffman; se requiere optimizar aún más los procesos tal como lo describe Abassi et al. (2023)

OpenMP es la técnica más usada debido a su interfaz legible. Sin embargo, la introducción de OpenMP en el código, es un desafío debido a los obstáculos en la gestión de la memoria compartida. Gonnord et. al (2022) en su publicación explican que el principal problema en la gestión de memoria es el no determinismo, el cual se produce cuando dos a más hilos en un único proceso acceden a la misma dirección de memoria concurrentemente.

En general los problemas presentados en la manipulación de imágenes en sistemas embebidos sugieren optimizar los procedimientos de la compresión en el algoritmo de Huffman. En ese sentido, en el presente trabajo se propone un esquema de paralelización. Mas aun teniendo en cuenta el problema del no determinismo, se presenta una propuesta hibrida a base de la combinación de librerías OpenMP y OpenMPI.

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

3.0 FUNDAMENTO TEORICO

La programación paralela

Hoy en día, la mayoría de las CPU contienen al menos dos núcleos y algunas tienen más de 50 unidades de procesamiento. Este esquema de multinúcleo lo podemos encontrar hasta en muchos dispositivos portátiles; además, sistemas más grandes como nodos de servidores, clusters y supercomputadoras ofrecen un grado aún mayor de paralelismo. Los usuarios necesitan escribir código altamente paralelizado para aprovechar al máximo las capacidades de cálculo de las arquitecturas de hardware actuales. Sin embargo, el diseño de algoritmos paralelos puede ser un desafío, especialmente para aquellos inexpertos, debido a las trampas comunes como los problemas al acceder concurrentemente a recursos compartidos, patrones de comunicación defectuosos que causan bloqueos, o la tarea no trivial de escalar eficientemente una aplicación sobre el número total de unidades de cálculo disponibles.

Si en un modelo donde cada entidad de procesamiento tiene solo acceso a su propia memoria local y se quiere acceder a los datos almacenados en otra entidad de procesamiento, será necesario implementar una comunicación explicita. A este tipo de arquitectura se conoce como sistema de memoria distribuida. Cada entidad de procesamiento está conectado a través de una red de interconexión. La interfaz de paso de mensajes MPI es la mas conocida en este tipo de arquitectura.

El otro modelo de arquitectura paralela es el de los sistemas de memoria compartida. En este caso todas las entidades de procesamiento pueden acceder a un espacio de memoria común a través de un bus común a todos. En adición a esta memoria compartida, cada entidad de procesamiento tiene una memoria local que funciona como cache para el acceso a la memoria principal el cual es muy costosa en recursos. Los datos almacenados en el cache deben ser coherentes con los de la memoria compartida. A esto se conoce como coherencia en cache. LA interfaz mas usada para este tipo de arquitectura es OpenMP.

(Schmidt, B et al. 2018)

En la programación paralela la finalidad es dividir el trabajo para realizarlo en los núcleos. Podemos mencionar dos enfoques: paralelismo en las tareas y paralelismo en los datos. En el primer enfoque el problema se divide en varias tareas las cuales serán distribuidas en los núcleos; mientras que en el segundo enfoque se divide los datos usados en resolver el problema para que sean procesados en los diferentes núcleos, y cada núcleo realiza casi las mismas operaciones con su correspondiente parte del dato.

En ese ámbito de tareas y procesamientos de datos de manera independiente, surge las siguientes necesidades:

a) La comunicación entre uno o más núcleos en la cual uno de ellos notifica su status.
b) La coordinación a través un correcto balanceo de carga en los núcleos, no permitiendo que algún núcleo termine de procesar mucho más anticipado que otro.

Tesis publicada con autorización del autor
No olvide citar esta tesis

c) La sincronización, el cual va permitir que los núcleos trabajen de manera cooperativa y no accedan recursos de memoria al mismo tiempo.

(Bernal F et al. 2018)

Por lo general existen extensiones en el lenguaje C y C++ que son construcciones paralelas explicitas para la programación en paralelo. Las extensiones más comunes son las librerías MPI (de sus siglas del inglés, Message Passing Interface) y las librerías OpenMP.

De manera general, sea cual fuese la arquitectura en la que se está paralelizando un problema, es necesario tener las siguientes consideraciones:

a) El particionamiento: Una determinada tarea necesita descomponerse en pedazos. Para ello existe muchas formas siendo las más comunes las que se emplean para el paralelismo de datos, paralelismo de tareas y paralelismo de modelo.

b) La comunicación: El esquema de particionamiento determina la cantidad y tipo de comunicación requerida entre procesos ó hilos.

c) La sincronización: Con el propósito de cooperar de una manera apropiada, los hilos o procesos necesitan ser sincronizados.

d) El balanceo de carga: La cantidad de trabajo necesita dividirse equitativamente entre los hilos ó procesos con el propósito de balancear la carga y minimizar los tiempos de hilos ó procesos ociosos.

El paralelismo de datos

En el paralelismo de datos, los datos se distribuyen a través de los diferentes procesadores o núcleos, los cuales pueden operar sobre sus datos asignados. El esquema de particionamiento también determina que comunicación es necesaria entre tareas. En este tipo de esquema se usa la arquitectura MPI y adicionalmente es necesario realizar sincronización entre procesos subsecuentes.

El paralelismo de tareas

Llamado también descomposición funcional por otro lado asigna diferentes operaciones a los procesadores los cuales están llevando a cabo sobre los mismos datos. Se centra en distribuir tareas, que son realizadas concurrentemente por procesos ó hilos, a través de diferentes procesadores. A diferencia de lo que ocurre en el paralelismo de datos, donde la misma tarea se ejecuta en diferentes componentes de datos, en el caso del paralelismo de tareas cada hilo de ejecución realiza muchas tareas diferentes de manera simultánea en los mismos datos. Un método muy usado de paralelismo de tareas en procesos a nivel de sistema operativo es el uso de tuberías, donde el procesamiento de un único conjunto de datos es realizado a través de una secuencia de tareas distintas, en este esquema cada una de las tareas se ejecuta de manera independiente respecto a las otras.

Tesis publicada con autorización del autor
No olvide citar esta tesis

A nivel de arquitectura en hardware con varios procesadores, para conseguir un paralelismo de tareas es necesario que cada procesador o núcleo ejecute un hilo o proceso diferente del conjunto de datos variado. Adicionalmente en algunos casos es necesario conseguir una comunicación entre sí entre los procesos mientras se ejecutan las tareas. En este caso esta comunicación se da por mensajería la cual forma parte del flujo en el cual interactúan los hilos o procesos.

El paralelismo de tareas destaca la característica distribuida del procesamiento ya que diferentes hilos o procesos van ejecutando diferentes partes del conjunto de datos. Mientras que el paralelismo de datos esta mas enfocado en los datos ya que una misma tarea se realizan en varios hilos o procesos. En la práctica, la mayoría de los programas comprende un espectro que va desde el paralelismo de tareas hasta el paralelismo de datos.

El paralelismo de modelo

El paralelismo de modelos es una técnica de computación paralela en la que un modelo de aprendizaje profundo se divide en varias partes, y cada parte se procesa en un dispositivo de hardware diferente. Esta técnica se utiliza principalmente para entrenar modelos grandes que no caben en la memoria de un solo dispositivo.

La programación con memoria distribuida (MPI)

Desde el punto de vista de un desarrollador, un sistema con memoria distribuida consiste de una colección de pares núcleo-memoria conectados por una red donde la memoria asociada a un determinado núcleo únicamente es accedida por dicho núcleo. El sistema con memoria distribuida más conocido emplea la transmisión de mensajes. En este modelo un programa que se ejecuta en un par núcleo-memoria generalmente se le denomina proceso, y dos procesos pueden comunicarse mediante la invocación a funciones. Así, un proceso invoca a una función send mientras que la otra invoca a una función receive. La implementación más utilizada de este modelo se llama MPI (por sus siglas de inglés Message-Passing Interface) (Gropp W, 2014)

MPI consiste de una librería de funciones que pueden ser invocadas desde programas en C, C++ y Fortran. Los comandos para compilar y ejecutar MPI dependen de la versión de implementación que se este usando. Por ejemplo, en OpenMPI se usa el comando mpic++ para compilar código escrito en el lenguaje C++ y mpirun acompañado del parámetro -np que indica el número de procesos para la ejecución de nuestro programa. Si uno de los procesos falla entonces toda la aplicación se detiene dando un código de error.

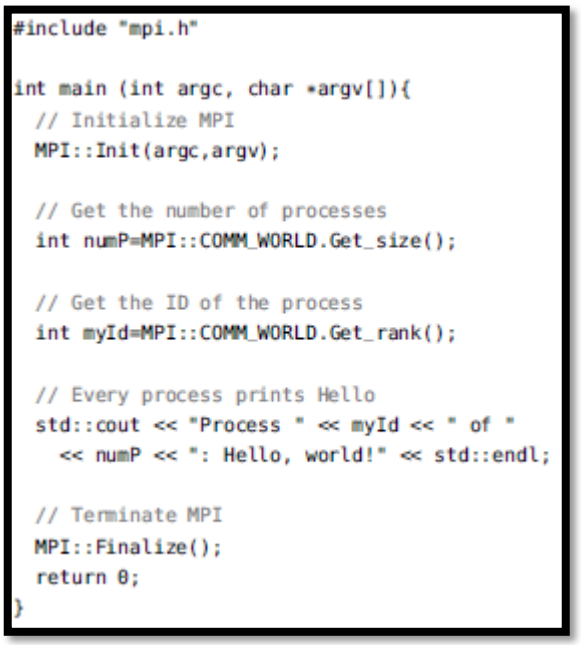
En una función principal main de un programa bajo MPI, todos los procesos son complemente independientes hasta que aparezca la instrucción MPI::Init(). A partir de esta inicialización, los procesos pueden colaborar, enviar/recibir mensaje ó sincronizar hasta alcanzar la instrucción MPI::Finalize(). Después de la finalización, se liberan todos los recursos asignados al MPI.

Tesis publicada con autorización del autor
No olvide citar esta tesis

MPI usa objectos comunicadores para definir que colección de procesos puede comunicarse con cada otros, la clase abstracta base es denominada MPI::Comm; así el objeto predefinido MPI::COMM_WORLD hace referencia a todos los procesos. Dentro de un objeto comunicador existe un identificador único asignado por el sistema cuando el proceso se inicializa. A esto se conoce como rank. En la siguiente figura se muestra un ejemplo de código MPI para un programa de Hola mundo.

Figura 1-Ejemplo de programa Hola mundo MPI

Ejemplo de código para programa Hola mundo en MPI.



```
#include "mpi.h"

int main (int argc, char *argv[]){
  // Initialize MPI
  MPI::Init(argc,argv);

  // Get the number of processes
  int numP=MPI::COMM_WORLD.Get_size();

  // Get the ID of the process
  int myId=MPI::COMM_WORLD.Get_rank();

  // Every process prints Hello
  std::cout << "Process " << myId << " of "
    << numP << ": Hello, world!" << std::endl;

  // Terminate MPI
  MPI::Finalize();
  return 0;
}
```

Fuente: Parallel Programming: Concepts and Practice por Schmidt (2018).

Por lo general los procesos intercambian mensajes y para ello el proceso send (envío) involucra almacenar el dato en un buffer y notificar al dispositivo de comunicación (que normalmente es la red) que la comunicación puede iniciarse. Sin embargo, el proceso destino todavía tiene que tener conocimiento de una rutina receive (recibir) que indica que desea recibir el dato. Una vez que esto sucede, el dato es transmitido y ambos procesos pueden continuar su trabajo. Puede ocurrir que los procesos no continúen hasta que la comunicación haya terminado el cual es el caso más común, en este caso se conoce como bloqueante, pero puede ocurrir que no se bloquee, en este caso es no bloqueante.

La programación con memoria compartida (OpenMP)

Desde el punto de vista de un desarrollador, un sistema con memoria compartida consiste de una colección de núcleos los cuales acceden a una memoria global, así cada núcleo puede acceder a cualquier ubicación de memoria. Es decir, en este sistema, cada hilo o proceso puede acceder a cualquier localización de memoria. OpenMP, muchas veces permite al programador indicar simplemente que un bloque de código debe ejecutarse en

Tesis publicada con autorización del autor
No olvide citar esta tesis

paralelo, y la determinación precisa de las tareas y qué subproceso debe ejecutarlas se deja en manos del compilador y del sistema en tiempo de ejecución.

OpenMP, permite que el compilador y el sistema en tiempo de ejecución determinen algunos de los detalles del comportamiento de los hilos, por lo que puede ser más sencillo codificar algunos procesos paralelos utilizando OpenMP. El coste es que algunas interacciones de hilos de más bajo nivel pueden ser difíciles de programar.

OpenMP fue desarrollado por un grupo de programadores que creían que escribir programas de alto rendimiento a gran escala utilizando APIs como Pthreads era demasiado difícil debido al tratamiento en bajo nivel para los hilos, y definieron la especificación OpenMP para que los programas de memoria compartida pudieran desarrollarse a un nivel superior. De hecho, OpenMP se diseñó explícitamente para permitir a los programadores paralelizar de forma incremental programas en serie existentes. Una de sus características mas importantes es la posibilidad de paralelizar muchos bucles for solamente haciendo pequeños cambios al código fuente (Quinn M, 2011).

La filosofía de OpenMP se basa en el uso de directivas de compilación llamadas pragmas para darle ciertas instrucciones al compilador sobre cómo se va paralelizar el código. Si el compilador no soporta dichas instrucciones pragmas simplemente las va ignorar. Las directivas son adicionadas al código para permitir comportamientos que no son parte de las especificaciones del compilador C o C++.

El número de hilos por defecto está determinado en tiempo de ejecución y es igual al número de núcleos de CPU lógicos vistos por el sistema operativo. Puede ser modificado definiendo la variable de entorno llamada OMP_NUM_THREADS. Mas aun los hilos utilizados puede ser especificados pueden ser especificados en el código fuente usando el comando set_num_threads().

Una de las directivas más comunes es #pragma omp parallel; la cual se utiliza para bifurcar y unir el hilo de ejecución real. Las instrucciones que se ejecutaran están en un bloque definido por llaves "{…}". En la siguiente figura se muestra un ejemplo de código OpenMP para un programa de Hola mundo.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Figura 2-Ejemplo de programa Hola mundo OpenMP

Ejemplo de código para programa Hola mundo en OpenMP.

```
#include <iostream>
#include <omp.h>

int main() {
    // run the block after the pragma in four threads
    #pragma omp parallel num_threads(4)
    {
        int i = omp_get_thread_num();
        int n = omp_get_num_threads();
        std::cout << "Hello world from thread "
                  << i << " of " << n << std::endl;
    }
}
```

Fuente: Parallel Programming: Concepts and Practice por Schmidt (2018).

Otra directiva importante es #pragma omp parallel for en OpenMP y es una herramienta poderosa para paralelizar bucles en programas escritos en lenguajes como C++ o Fortran. La directiva actúa como divisora de un conjunto de "n" índices en un bucle for en pedazos de tamaño aproximado de n/p donde p es el número de hilos utilizados. Se debe tener en cuenta que OpenMP no transforma mágicamente u bucle for secuencial en uno paralelo, sino que se debe tener consideraciones teóricas que garantizan la correcta transformación. En conclusión, esta directiva permite i) dividir los índices de un bucle for eventualmente en pedazos y ii) procesar subsecuentemente cada pedazo de manera independiente. Adicionalmente cada hilo puede acceder cualquier variable ó arreglo declarado fuera del ámbito de paralelización, esto debido a que OpenMP es un framework operando en arquitecturas de memoria compartida. Aquí se tiene que garantizar que tu código esté libre de condiciones de carrera. Las condiciones de carrera ocurren cuando dos o más hilos o procesos intentan acceder y modificar al mismo tiempo datos compartidos por dos o más procesos, esto puede conducir a resultados que pueden ser incorrectos para alguno de los procesos involucrados.

La compresión de datos

Con el apogeo actual de los medios digitales tales como el crecimiento de internet, el desarrollo explosivo de las comunicaciones móviles y el desarrollo de las comunicaciones por medio de videos; resulta inevitable indicar que la compresión de datos juega un papel importante en toda esa evolución de los medios digitales. No seria posible subir imágenes, audio y video en los sitios web si no existiesen los algoritmos de compresión de datos. La televisión digital no sería posible sin considerarse involucrado la compresión de datos. La compresión de datos permite representar la información en una manera compacta. Esta representación compacta se crea identificando y usando estructuras que existen en los datos. La razón de la importancia de comprimir datos es que cada vez generamos más y más información de forma digital. Por ejemplo, según indica Sayood (2018); en un

Tesis publicada con autorización del autor
No olvide citar esta tesis

segundo de video sin compresión usando el formato CCIR 601 se necesitan mas de 20 megabytes o 160 megabytes. Si consideramos la duración de una película entonces debemos pensar en comprimirla. Para representa 2 minutos de música no comprimida y de buena calidad en CD es necesario de 84 millones de bits. Descargarlo de internet tomaría mucho tiempo.

Quizá una de las primeras ideas acerca de la compresión de datos proviene del siglo 19 con la creación del código Morse para enviar letras usando puntos y líneas. Samuel Morse se dio cuenta que ciertas letras ocurrían con mayor frecuencia que otras. Con el propósito de reducir el tiempo promedio en enviar un mensaje; asigno una secuencia corta a letras que ocurrían con mayor frecuencia tales como e (.) y a (. -), para las que ocurrían con menor frecuencia asigno una secuencia larga como en el caso de q (− − · −) and j (· − − −). En este caso, se está usando una estructura estadística para comprimir los datos, pero no es la única estructura que se puede encontrar en los diferentes tipos de datos. Por ejemplo, en el caso de la voz, se puede utilizar el contenido espectral de la voz para poder comprimirla.

Como mencionamos anteriormente, comprimir un dato es representar lo mismo, pero de una manera mas compacta. Desde este punto de vista al recuperar el dato después de la compresión, este debería ser idéntico al original. Bajo este contexto a veces resulta apropiado omitir ciertas características de la información que no es perceptible al sentido humano. Por ejemplo, si se trata de información relacionada a imágenes, hay ciertas tonalidades que no son perceptible al ojo humano por lo que daría lo mismo recuperar la información y que no sea totalmente idéntica a la original. Bajo este punto de vista los tipos de compresión se clasifican en: compresión sin perdidas y compresión con perdida.

Compresión de datos sin perdida

Como su nombre lo indica, en esta técnica la información recuperada es exactamente igual a la original. Este tipo de técnica se utiliza en aplicaciones que no toleran que exista alguna diferencia entre la información original y la recuperada desde la compresión. Por ejemplo, en el análisis de imágenes radiológicas es de suma importancia que la imagen procesada comprimiéndola sea idéntica al original para que no confunda gravemente al radiólogo. Otro ejemplo se da en las imágenes satelitales, a menudo, los datos recopilados por satélites se procesan posteriormente para obtener indicadores numéricos relacionados con la vegetación, la deforestación, etc. Si los datos reconstruidos no son idénticos a los originales, cualquier procesamiento posterior podría realzar las diferencias. En algunos casos, no es posible volver a obtener los mismos datos. Por lo tanto, no es recomendable permitir que aparezcan diferencias en el proceso de compresión inicial.

Compresión de datos con perdida

En esta técnica de compresión, los datos que se recuperan no son idénticos al original. Sin embargo, esa distorsión a veces resulta aceptable. Generalmente en esta técnica se obtiene mayores ratios de compresión que los obtenidos en las técnicas sin perdida de información. Por ejemplo, si estamos almacenando o transmitiendo voz se puede tolerar

Tesis publicada con autorización del autor
No olvide citar esta tesis

cierta distorsión la cual es suficiente para cierta ocasión. Otro ejemplo similar es cuanto tratamos una secuencia de video donde la reconstrucción no es idéntica a la original pero esta tolerancia no es perceptible.

ALGORITMO DE HUFFMAN

Uno de los algoritmos más conocidos y que corresponde con técnicas de compresión sin perdida de información es el algoritmo de Huffman. Este algoritmo de huffman es código de longitud variable, donde la idea del algoritmo es recodificar los bytes de cada información con secuencias más cortas. Se dice que es de longitud variable porque la longitud de cada código depende de la frecuencia (absoluta o relativa) de aparición de cada byte de información (Sznajdleder 2012). Tal es así, que cuanto más frecuente sea un símbolo, su código asociado será más corto. De esta manera la codificación de Huffman se torna primero en un modelo estadístico para conocer la frecuencia de cada elemento byte en la información.

Según la frecuencia de aparición de los bytes, se va construyendo una estructura de datos llamada árbol binario. Según Sznajdleder (2012), el algoritmo de construcción del árbol puede resumirse así:

1.- Crear un nodo hoja para cada símbolo, asociando un peso según su frecuencia de aparición e insertarlo en una lista ordenada ascendentemente.

2.- Mientras haya más de un nodo en la lista:

a) Eliminar de la lista los dos nodos con menor frecuencia

b) Crear un nuevo nodo interno que enlace a los nodos anteriores, asignándole como peso la suma de los pesos de los nodos hijos y etiquetamos la arista del nodo derechos con un uno y del nodo izquierdo con un cero.

c) Insertar el nuevo nodo en la lista, en el lugar que corresponda según el peso.

3.- El nodo que queda es el nodo raíz del árbol

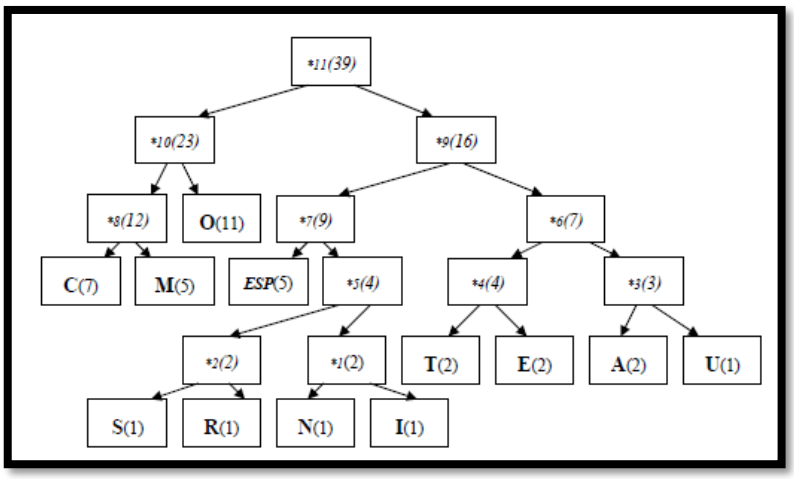
A continuación, se muestra en la figura 1 un ejemplo de cómo finalmente quedaría un árbol binario.

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Figura 3-Ejemplo de árbol binario

Ejemplo de árbol binario obtenido.



Fuente: Algoritmos a Fondo - Con implementaciones en c y java por Sznajdleder (2012).

Una vez construido el árbol binario, se procede con la asignación de los códigos huffman para cada carácter, los cuales se hacen siguiendo el camino de la hoja que representa el carácter. El camino implica concatenar ceros o unos dependiendo del camino.

Finalmente, en el proceso de compresión se reemplaza cada carácter de la información por su respectivo código huffman. De esta manera, aquellos caracteres que más se repitan tendrán un código más corto. (Sznajdleder 2012)

4.0 JUSTIFICACION DE LA INVESTIGACION

Variable independiente

Según Sampieri, et al. (2014), la variable independiente está relacionada con los causantes de la investigación.

En la presente investigación, la variable independiente es la librería hibrida OpenMPI/OpenMP

Variable dependiente

Según Sampieri, et al. (2014), la variable dependiente son los efectos o consecuentes de la investigación.

En la presente investigación, la variable dependiente es el procesamiento de imágenes a través de la compresión de la imagen

Tesis publicada con autorización del autor
No olvide citar esta tesis

Justificación teórica

El presente proyecto permitirá mejorar la técnica de compresión de la información usando el algoritmo de Huffman, el cual es muy útil en el procesamiento de imágenes; y a través del empleo de una librería hibrida OpenMPI/OpenMP reforzando la utilización del paralelismo en lugar de que estas técnicas se trabajen de manera independiente.

Justificación practica

La aplicación de este proyecto se enfoca en los sistemas embebidos en donde los recursos son bastante limitados y donde es necesario la optimización de los procesos usando el paralelismo. Por otro lado, el estudio puede contribuir a manera de recomendación para problemas de similar naturaleza.

5.0 OBJETIVOS DE LA INVESTIGACION

Objetivo general

El objetivo general de la investigación es determinar la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman

Objetivos Específicos 1

Determinar la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman.

Objetivos Específicos 2

Determinar la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el tamaño del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman.

6.0 HIPOTESIS DE LA INVESTIGACION

Hipótesis General

La librería con tecnología hibrida OpenMPI/OpenMP mejora el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman

Hipótesis Especifica 1

La librería con tecnología hibrida OpenMPI/OpenMP mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Hipótesis Especifica 2

La librería con tecnología hibrida OpenMPI/OpenMP mejora el tamaño de compresión del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman.

7.0 METODOS Y MATERIALES

La presente investigación tiene un enfoque cuantitativo ya que es necesario realizar mediciones a la experiencia. El diseño es de tipo experimental ya que el objetivo general es determinar la influencia de una librería con tecnología hibrida OpenMPI/OpenMP sobre el procesamiento de imágenes al comprimir la información usando el algoritmo de Huffman. En esta investigación, se toman mediciones antes de la aplicación de la librería hibrida y luego una medición posterior por lo que corresponde a la clase preexperimental del diseño experimental ya que este se caracteriza porque hay una medición previa a la aplicación del estímulo y finalmente una medición posterior a dicho estimulo según Sampieri et al. (2014). En este diseño preexperimental se ha aplicado la preprueba/posprueba sobre un mismo grupo de estudio. El enfoque cuantitativo permite derivar una o más hipótesis que serán sometidas a prueba mediante el diseño de investigación. A final se discute y se llega a conclusiones de como los resultados concuerda con los conocimientos existentes.

En la presente investigación, la variable independiente es la librería hibrida OpenMPI/OpenMP y la variable dependiente es la compresión en el procesamiento de imágenes.

Dimensiones a considerar en la variable dependiente

Las dimensiones de la variable dependiente están relacionadas con las siguientes características: el tiempo de ejecución y el tamaño del archivo.

Indicadores a considerar en la variable dependiente

Los indicadores específicos a considerar para la variable dependiente según la dimensión del tiempo de ejecución son:

a) Cantidad de hilos de procesamiento

b) Tiempo de ejecución inicial

c) Tiempo de ejecución final

Los indicadores específicos a considerar para la variable dependiente según la dimensión del tamaño del archivo son:

a) Rango de compresión

b) Tamaño de archivo inicial

Tesis publicada con autorización del autor
No olvide citar esta tesis

c) Tamaño de archivo final

El instrumento para la recolección de datos en la presente investigación serán las fichas de observación para el tiempo de ejecución y el tamaño de archivo (véase anexo 3). Por lo que la técnica está basada en la observación y se elaborarán fichas de observación en una condición inicial antes de la implementación de la librería hibrida y en una condición posterior considerando el paralelismo con la librería hibrida. En estas fichas se considerará los siguientes indicadores: tiempo de ejecución, tamaño inicial del archivo, tamaño final del archivo y cantidad de hilos de procesamiento.

Confiabilidad del instrumento de medición

Según Sampieri, et al. (2014), toda medición está sujeta a errores. La confiabilidad de las fichas de observación estará determinada por el método de la estabilidad en la medición. Es decir que el proceso de medición se va realizar más de dos veces con el propósito de encontrar una correlación en los valores obtenidos; esto garantizara la confiabilidad del instrumento. Las condiciones bajo las cuales se realizan la experiencia no se alteran ya que la constitución de las imágenes no cambia en el tiempo. Así para la confiabilidad por test-retest se realiza una repetición de las pruebas hasta en dos oportunidades a un grupo de 30 imágenes en diferentes momentos, realizaremos un cálculo de correlación entre dichas pruebas con el objeto de encontrar el coeficiente de correlación Pearson. Consideramos que un coeficiente de correlación de Pearson mayor a 0.80 indica una correlación considerable según la interpretación que da Sampieri et al. (2014). En la presente investigación se tuvo un coeficiente de Pearson igual a 0.989 para las mediciones en el caso de la preprueba y de 0.978 para las mediciones en el caso de la posprueba. Lo cual es un valor aceptable para considerar la confiabilidad.

En cuanto a la validez del instrumento, se ha revisado trabajos anteriores y todos ellos consideran que el tiempo de ejecución y el tamaño del archivo determinan en gran medida el desempeño de un proceso de compresión de imágenes. Adicionalmente el tiempo de ejecución, la cantidad de hilos de procesamiento y el rango de compresión están estrechamente relacionados al objeto de la presente investigación y pueden ser útiles para inferir conclusiones.

La población objetivo para este estudio lo constituye una totalidad de 30 imágenes satelitales cuyo tamaño va desde 6.77 MB hasta 1.34 GB. Las imágenes satelitales fueron obtenidas del Sistema de Información y Datos del Sistema de Observación Terrestre (EOSDIS en inglés), se seleccionaron al azar correspondiendo a regiones diferentes y a diferentes tiempos de exposición. Las imágenes están conformadas por archivos con tamaño diferente con el propósito de tener diferentes participantes. Esto garantiza que no exista sesgo en la selección de la muestra según Bhandari (2023) en su artículo relacionado con la asignación aleatoria. Dado que la población es finita y de tamaño reducido, no se ha realizado un cálculo muestral. En su lugar, se incluyó a todas las imágenes que comprenden la población.

Criterios de Selección:

- Imágenes de diferentes regiones y tiempo de exposición.
- Tamaño de imagen en el intervalo de 6.8 MB hasta 1.3 GB

Tesis publicada con autorización del autor
No olvide citar esta tesis

- Imagen satelital sin comprimir.

Aunque se examina a toda la población, aplicamos los principios estadísticos para analizar los datos y realizamos pruebas estadísticas para responder a las preguntas de investigación específicas.

Se realiza un comparativo entre los escenarios de un antes y un después de emplear la librería. Primero se estudia la normalidad de la diferencia de las mediciones de los datos según las dimensiones de estudio para ver si se da un enfoque paramétrico o no paramétrico. En este análisis preliminar si usara un nivel de confianza de 95% y para este caso de prueba de normalidad en muestras pequeñas se emplea la prueba de Shapiro Wilk. Finalmente se tiene el criterio de decisión según el P-valor (es decir el grado de significancia en el análisis).

Si después del análisis se determina que no hay normalidad en los datos entonces aplicaremos el test de Wilcoxon ya que sería un caso no paramétrico para validar la hipótesis de que la librería hibrida mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman y también mejora el tamaño del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman. Se trabajará con un nivel de confianza de 95% y un margen de error de 5%.

Si, por otro lado, el análisis preliminar determina que existe un comportamiento normal entonces utilizaremos la prueba t de student para grupos relacionados. Con esa información validaremos la hipótesis de que la librería con tecnología hibrida OpenMPI/OpenMP mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman y también validaremos la hipótesis de que la librería con tecnología hibrida OpenMPI/OpenMP mejora el tamaño del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman.

Metodología para el desarrollo

El método propuesto va permitir la creación de una librería hibrida OpenMP/OpenMPI optimizada, la cual va poder ser integrada. En la figura 4 se muestra de manera general las fases en la metodología a seguir.

Figura 4-Fases de la metodología

Fases de la metodología a seguir



El desarrollo de la librería está basado en OpenMP y OpenMPI que ha sido compilado con el compilador g++ bajo la envolvente mpic. El desarrollo se realizó en una computadora con procesador AMD modelo A9-9420, con tarjeta gráfica integrada Radeon R5 y dos núcleos. Con una memoria RAM de 8GB y disco duro de 300GB en un entorno ubuntu-20.04-desktop-amd64.

Tesis publicada con autorización del autor
No olvide citar esta tesis

1.- Definición de funcionalidad

En la presente investigación se tomará en cuenta las funcionalidades importantes en MPI tales como la mensajería explicita, la sincronización implícita, y nada compartido con la arquitectura. Combinadas con las funcionalidades de OpenMP tales como la mensajería implícita, la sincronización explicita y todo compartido con la arquitectura.

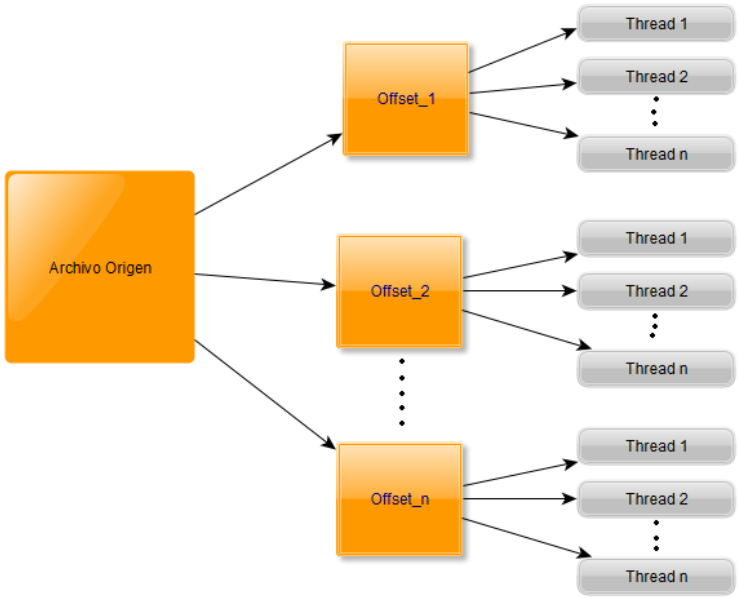
Se lee el archivo imagen considerando una cantidad de offset según el número de núcleos obtenidos, en este procedimiento se hace uso de una arquitectura con paralelismo de datos el cual se trató en el fundamento teórico. Posteriormente cada núcleo empleara una cantidad de hilos definida para invocar las funcionalidades OpenMP, considerando aquí una arquitectura con paralelismo de tareas. Finalmente, el procesamiento individual de los núcleos va generar de manera ordenada al archivo comprimido.

Las directivas estarán encapsuladas y en una primera aproximación, se aplicará a los algoritmos de compresión de datos en los que no se presentan perdidas.

En la siguiente figura 5, se esquematiza las funcionalidades principales de la librería.

Figura 5-Esquema de funcionalidades de la librería

Esquema de las funcionalidades basicas de la libreria



2.- Creación de funciones

Se considerarán funciones que permitan inicializar, invocar y ejecutar operaciones como la compresión de datos con poca perdida de información. Además de poder realizar la multiplicación de matrices.

3.- Creación de la librería

Para poder escribir la librería, analizaremos el código fuente en C++ que se ejecuta de manera secuencial. Luego, se tratará de paralelizar las secciones de código que sean apropiadas. Finalmente, se construirá el código paralelizado y optimizado. El

Tesis publicada con autorización del autor
No olvide citar esta tesis

procedimiento se repetirá en el algoritmo seleccionado en el presente trabajo. Los códigos fuente se compilarán con el compilador g++ bajo la envolvente mpic.

4.- Pruebas de la librería

Las pruebas se van a realizar en una computadora con procesador AMD modelo A9-9420 con tarjeta gráfica integrada Radeon R5 y dos núcleos. Con una memoria RAM de 8GB y disco duro de 300GB en un entorno ubuntu-20.04-desktop-amd64.

Se procederá con la creación de una librería hibrida OpenMP/MPI de tal manera que se pueda reducir el tiempo de ejecución al procesar los algoritmos de compresión de datos. El algoritmo a evaluar es de Huffman , se ha seleccionado este algoritmo ya que presentan la característica de no perder la calidad en la información al momento de comprimir.

5.- Evaluación del rendimiento

Se realizará un comparativo entre los escenarios de un antes y un después de emplear la librería. Las dimensiones que se van a analizar son el tiempo de ejecución y el tamaño del archivo. Se llevará a cabo la prueba estadística para las mediciones en las 30 imágenes satelitales. Se trata de grupos relacionados, es decir se evalúa en situaciones diferentes (un antes y un después). Primero se estudiarán la normalidad de la diferencia de las mediciones de los datos según las dimensiones de estudio para ver si se da un enfoque paramétrico o no paramétrico. En este análisis preliminar se usara un nivel de confianza de 95% y para este caso de prueba de normalidad en muestras pequeñas se usará la prueba de Shapiro Wilk. Finalmente se tiene el criterio de decisión según el P-valor (es decir el grado de significancia en el análisis).

Si después del análisis se determina que no hay normalidad en los datos entonces aplicaremos el test de Wilcoxon ya que sería un caso no paramétrico para validar la hipótesis de que la librería hibrida mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman y también mejora el tamaño del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman. Se trabajará con un nivel de confianza de 95% y un margen de error de 5%.

Si, por otro lado, el análisis preliminar determina que existe un comportamiento normal entonces utilizaremos la prueba t de student. Con esa información validaremos la hipótesis de que la librería con tecnología hibrida OpenMPI/OpenMP mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman y también validaremos la hipótesis de que la librería con tecnología hibrida OpenMPI/OpenMP mejora el tamaño del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman.

8.0 RESULTADOS

Para obtener los resultados, la recolección de los datos se realizó en dos etapas; el primer grupo (preprueba) tratándose de muestras que se van a comprimir sin considerar la librería hibrida y el segundo grupo (posprueba) haciendo uso de la librería hibrida OpenMPI y

Tesis publicada con autorización del autor
No olvide citar esta tesis

OpenMP, así mismo se realizó una comparación entre los resultados de ambas pruebas. Dentro de las pruebas estadísticos se siguieron varios pasos para abordar los indicadores clave de la variable de procesamiento de imagen a través de la compresión.

Pruebas de normalidad de los datos

Siendo el caso de una muestra pequeña, se utilizará la prueba de Shapiro Wilk para evaluar si la diferencia de tiempos entre la preprueba y la posprueba tiene un comportamiento normal. La evaluación se realizó a través de rutinas en Python para los casos del tiempo de ejecución y el tamaño de archivo. A continuación, en la siguiente figura se muestra el fragmento usado.

Figura 6-Fragmento en Python para el cálculo p-value respecto a al tiempo de ejecución Shapiro-Wilk

```
import scipy.stats as stats
import matplotlib.pyplot as plt

# Tiempos de compresión de imágenes en segundos
data = [2.22273,2.031942,1.5988,1.88435,3.42352,2.32487,2.25273,2.39649,
2.58826,3.43587,2.57282,2.30282,3.72087,4.42543,2.45509,1.01202,1.7611,
2.89738,1.96605,1.670482,1.95341,1.95847,1.89691,2.02293,1.944384,1.94073,
1.93524,3.20216,9.1718,153.9588]

# Realizamos la prueba de Shapiro-Wilk
shapiro_test = stats.shapiro(data)
print(f"Shapiro test statistic: {shapiro_test[0]}, p-value: {shapiro_test[1]}'
```

Figura 7-Fragmento en Python para el cálculo p-value respecto al tamaño de archivo Shapiro-Wilk

```
import scipy.stats as stats
import matplotlib.pyplot as plt

# Tiempos de compresión de imágenes en segundos
data = [29427,3046,29102,25798,22486,23224,22405,23330,22367,22509,23051,
14819,6095,8033,10114,7293,11913,21691,5057,10896,10088,3998,6960,8468,
163898,580189,2894,8257,2894928,134071913]

# Realizamos la prueba de Shapiro-Wilk
shapiro_test = stats.shapiro(data)
print(f"Shapiro test statistic: {shapiro_test[0]}, p-value: {shapiro_test[1]}'
```

El valor obtenido de P-value fue de 0.209 para el caso de las mediciones de tiempo de ejecución y 0.187 para el caso de las mediciones del tamaño de archivo; lo cual indica un valor mayor a 0.05 en ambos casos. Teniendo como hipótesis nula que la muestra presenta un comportamiento normal entonces de acuerdo al valor obtenido podemos concluir que si existe un comportamiento normal de la muestra para ambos casos.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Visualizando los datos en un gráfico Q-Q o grafico Cuantiles-Cuantiles:

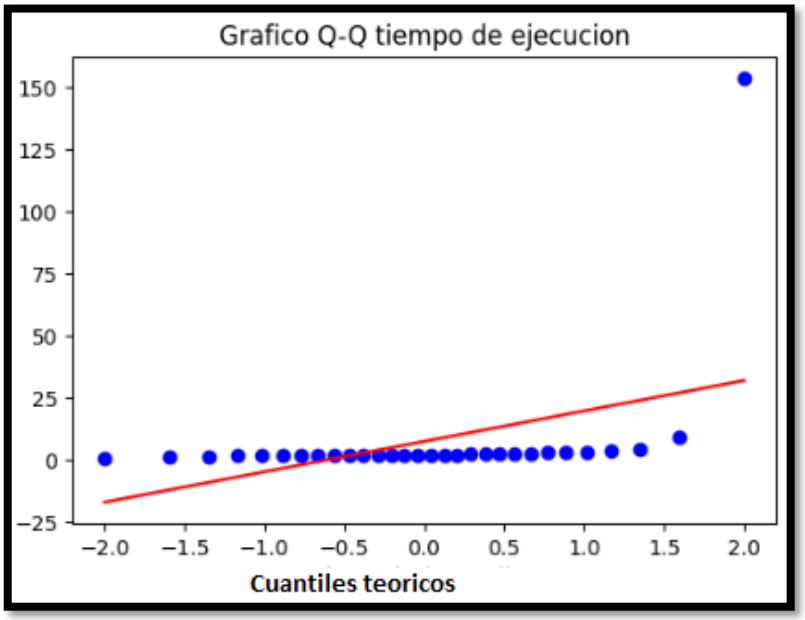
Figura 8-Fragmento en Python para visualizar el grafico Q-Q respecto al tiempo de ejecución

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

data = [2.22273,2.031942,1.5988,1.88435,3.42352,2.32487,2.25273,2.39649
    ,2.58826,3.43587,2.57282,2.30282,3.72087,4.42543,2.45509,1.01202,1
    .7611,2.89738,1.96605,1.670482,1.95341,1.95847,1.89691,2.02293,1
    .944384,1.94073,1.93524,3.20216,9.1718,153.9588]

plt.figure(figsize=(6,4))
stats.probplot(data, dist="norm", plot=plt)
plt.title("Grafico Q-Q tiempo de ejecucion")
plt.show()
```

Figura 9-Grafico Q-Q para las diferencias entre preprueba y posprueba en el caso de la medida del tiempo de ejecución



Tesis publicada con autorización del autor
No olvide citar esta tesis

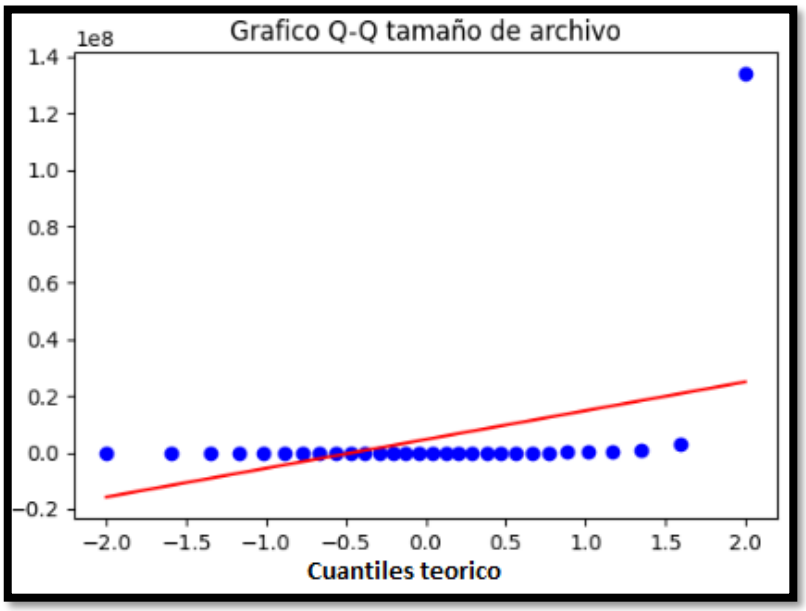
Figura 10-Fragmento en Python para visualizar el grafico Q-Q respecto al tamaño de archivo

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

data = [29427,3046,29102,25798,22486,23224,22405,23330,22367,22509
    ,23051,14819,6095,8033,10114,7293,11913,21691,5057,10896,10088,3998
    ,6960,8468,163898,580189,2894,8257,2894928,134071913]

plt.figure(figsize=(6,4))
stats.probplot(data, dist="norm", plot=plt)
plt.title("Grafico Q-Q tamaño de archivo")
plt.show()
```

Figura 11-Grafico Q-Q para las diferencias entre preprueba y posprueba en el caso de la medida del tamaño de archivo



Para crear el gráfico Q-Q, ordenamos nuestros datos en orden ascendente, dividimos los datos en cuantiles y luego graficamos nuestros cuantiles observados contra los cuantiles calculados a partir de una distribución normal teóricamente perfecta. El número de cuantiles de la distribución normal teórica se selecciona para coincidir con el tamaño de nuestros datos de muestra.

Si los datos tienen una distribución normal, los puntos en un gráfico Q-Q se ubicarán aproximadamente en una línea diagonal recta. Por el contrario, cuanto más se desvíen los puntos en el gráfico de una línea diagonal recta, es menos probable que el conjunto de datos tenga una distribución normal.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Por lo tanto, en el grafico Q-Q normal se visualiza que los datos se distribuyen con cierta característica normal ya que los puntos siguen aproximadamente una línea recta diagonal. Los datos no forman una curva que se desvía notablemente de una línea recta.

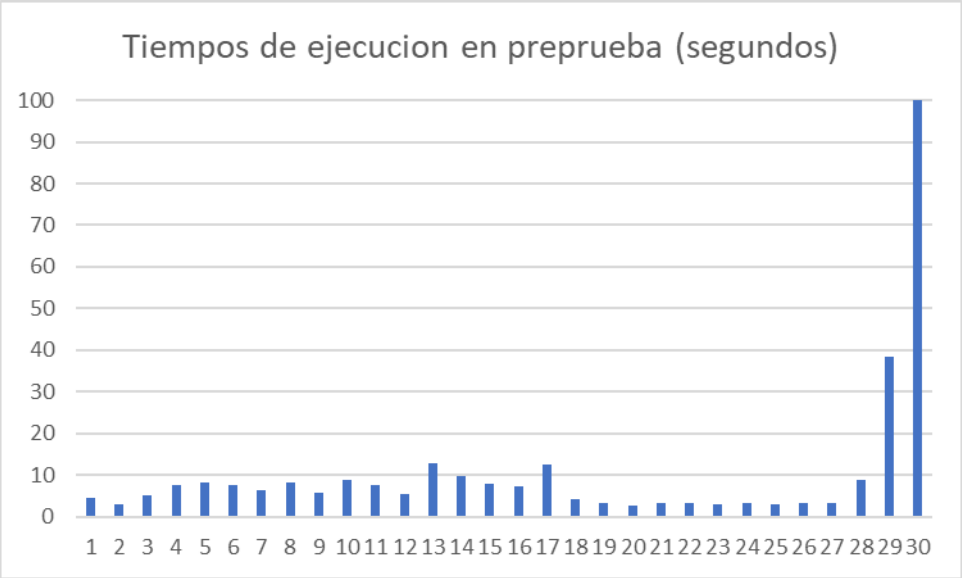
Pruebas estadísticas realizadas

Dado que se ha comprobado que las mediciones realizadas presentan un comportamiento normal entonces se utilizara la prueba t-Student para evaluar los cambios observados en la experiencia de acuerdo a las siguientes dimensiones:

- El tiempo de ejecución:
  Prueba de t de Student para comparar los tiempos de ejecución entre el grupo preprueba y posprueba. Se va determinar si la librería influyó significativamente en la mejora del tiempo de ejecución.
- El tamaño del archivo:
  Prueba de t de Student para comparar el tamaño de los archivos obtenidos entre el grupo preprueba y posprueba. Se va determinar si la librería influyo significativamente en el tamaño de los archivos al realizar el proceso de compresión.

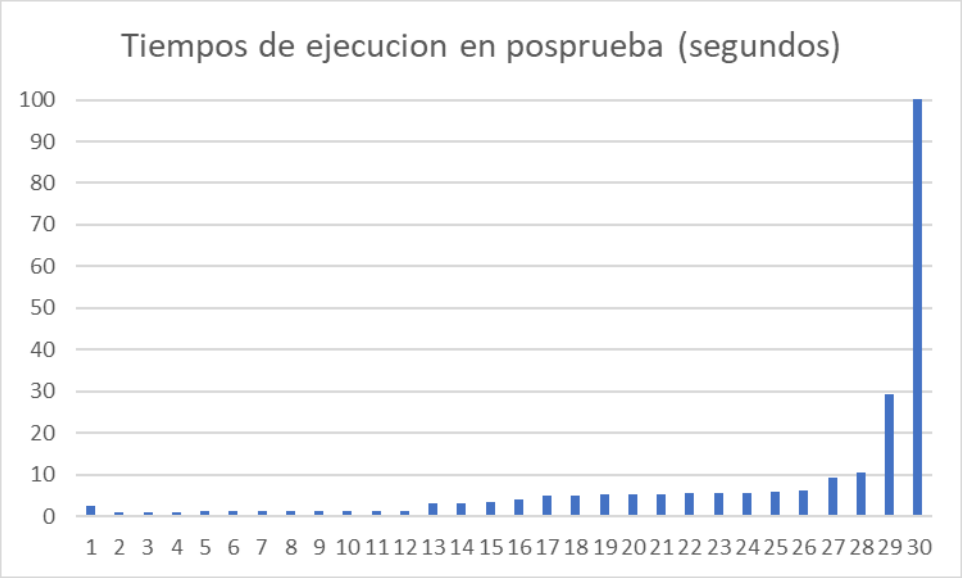
Dimensión: Tiempo de ejecución

Figura 12-Grafico tiempos de ejecución en preprueba



Tesis publicada con autorización del autor
No olvide citar esta tesis

Figura 13-Grafico tiempos de ejecución en posprueba



Prueba de t de Student para el tiempo de ejecución

Se obtuvo datos sobre los tiempos de ejecución del grupo de preprueba y el otro grupo en la posprueba.

Tabla 1-Datos estadísticos para el cálculo t student con el tiempo de ejecución

Grupo preprueba: n_1	30 imágenes, con tiempos de ejecución sin utilizar la librería
Promedio M_1	28.97 segundos
Desviación Estándar SD_1	45.89
Grupo Post-Test : n_2	30 imágenes, con tiempos de ejecución utilizando la librería
Promedio M_2	11.60 segundos
Desviación Estándar SD_2	30.02
α (Nivel de significancia)	0.05
Grado de Libertad $df = n_1 + n_2 - 2$	$df = 58$

Hipótesis Nula (H0)

H0: No hay diferencia significativa en los tiempos de ejecución entre preprueba y posprueba.

Hipótesis Alternativa (H1)

H1: Existe una diferencia significativa en los tiempos de ejecución entre preprueba y posprueba.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Procedimiento:

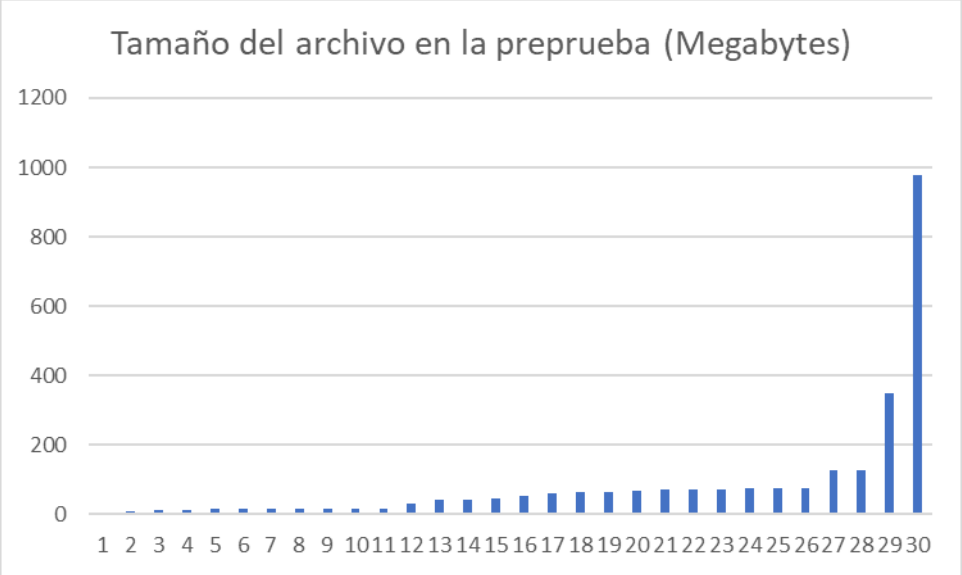
Ahora realizaremos la prueba t de Student. La fórmula para la prueba t es la siguiente:

$$t = \frac{(M_1 - M_2)}{\sqrt{\frac{SD_1^2}{n_1} + \frac{SD_2^2}{n_2}}} = t = \frac{(28.97 - 11.60)}{\sqrt{\frac{45.89^2}{30} + \frac{30.02^2}{30}}} = 1.735$$

Se compara el valor de t_calc con el valor crítico de la tabla de t de Student para determinar si hay una diferencia significativa; para un nivel de significancia del 0.05 y 58 grados de libertad, el valor de t en tablas para esas condiciones es 1.67. Dado que nuestro valor calculado de t (1.735) es ligeramente mayor que el valor crítico, podemos rechazar la hipótesis nula y concluir que hay una diferencia entre los tiempos de ejecución en la preprueba y posprueba. Los resultados de la prueba t indican que hay evidencia estadística para sugerir que el empleo de la librería ha tenido un impacto en la reducción del tiempo de ejecución al comprimir el archivo.

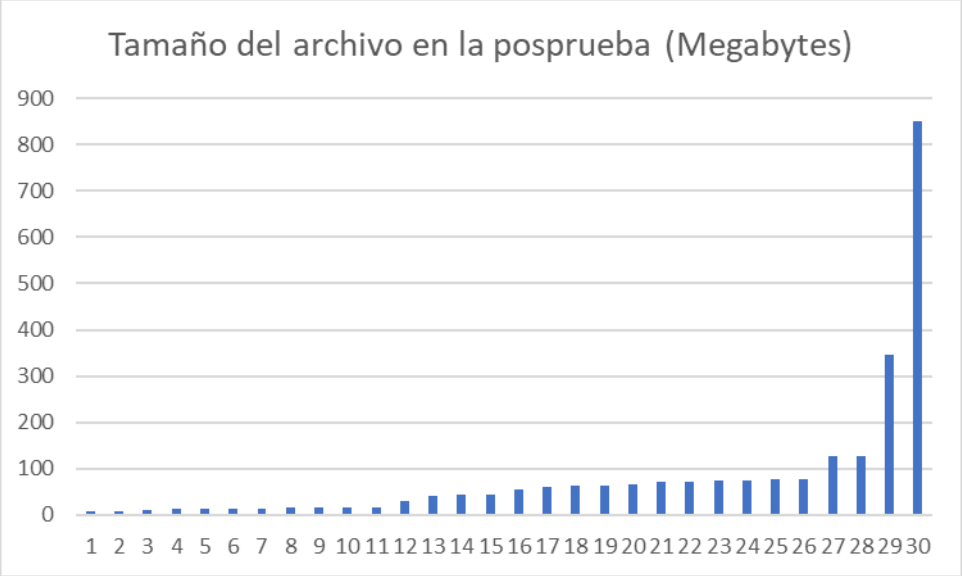
Dimensión: Tamaño del archivo

Figura 14-Grafico tamaño de archivo comprimido en preprueba



Tesis publicada con autorización del autor
No olvide citar esta tesis

Figura 15-Grafico tamaño de archivo comprimido en posprueba



Prueba de t de Student para el tamaño de archivo

Se obtuvo datos sobre los tamaños de archivo del grupo de preprueba y el otro grupo de la posprueba.

Tabla 2-Datos estadísticos para el cálculo t student con el tamaño de archivo

Grupo preprueba: n_1	30 imágenes, con tamaños diferentes comprimidos sin utilizar la librería
Promedio M_1	91912080.71 bytes
Desviación Estándar SD_1	188997941.5
Grupo Post-Test : n_2	30 imágenes, con tamaños diferentes comprimidos utilizando la librería
Promedio M_2	87308939.07 bytes
Desviación Estándar SD_2	166201609.8
α (Nivel de significancia)	0.05
Grado de Libertad $df = n_1 + n_2 - 2$	$df = 58$

Hipótesis Nula (H0)

H0: No hay diferencia significativa en los tamaños finales del archivo comprimido entre preprueba y posprueba.

Hipótesis Alternativa (H1)

H1: Existe una diferencia significativa en los tamaños finales del archivo comprimido entre preprueba y posprueba.

Tesis publicada con autorización del autor
No olvide citar esta tesis

Procedimiento:

Ahora realizaremos la prueba t de Student. La fórmula para la prueba t es la siguiente:

$$t = \frac{(M_1 - M_2)}{\sqrt{\frac{SD_1^2}{n_1} + \frac{SD_2^2}{n_2}}} = t = \frac{(91912080.71 - 87308939.07)}{\sqrt{\frac{188997941.5^2}{30} + \frac{166201609.8^2}{30}}} = 0.100$$

Se compara el valor de t_calc con el valor crítico de la tabla de t de Student para determinar si hay una diferencia significativa; para un nivel de significancia del 0.05 y 58 grados de libertad, el valor de t en tablas para esas condiciones es 1.67. Dado que nuestro valor calculado de t (0.100) es mucho menor que el valor crítico, podemos aceptar la hipótesis nula y concluir que no hay una diferencia significativa entre los tamaños de los archivos finales comprimidos en la preprueba y posprueba. Los resultados de la prueba t indican que no hay evidencia estadística para sugerir que el empleo de la librería ha tenido un impacto significativo en la variación del tamaño del archivo comprimido.

9.0 DISCUSION

En primer lugar, se ha demostrado que la experiencia de medir los tiempos de ejecución y el tamaño de archivos para el proceso de la compresión de archivos de imágenes satelitales usando el algoritmo de huffman, está relacionado a eventos con distribución normal; corroborándose esto con la característica continua de los datos. La librería hibrida ha brindado una solución apropiada para mejorar el tiempo de compresión de archivos de imágenes a través del algoritmo de Huffman. Sin embargo, no fue así para el caso del tamaño del archivo; observándose que no existe una modificación significativa del tamaño final del archivo de imagen al utilizar la librería hibrida. A partir del resultado obtenido durante la investigación, se contrastó la hipótesis de que la librería con tecnología hibrida OpenMPI/OpenMP mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman. Por otro lado, la hipótesis de que la librería con tecnología hibrida OpenMPI/OpenMP mejora el tamaño de compresión del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman no fue significativa teniendo en cuenta su análisis bajo la prueba t Student.

El bajo rendimiento de la librería hibrida respecto al ratio de compresión, es decir a la relación entre tamaño de archivo imagen final respecto del inicial, se debe principalmente a la naturaleza estadística del algoritmo y a la elección de los prefijos para codificar los caracteres propios de la estructura en árbol; haciendo que la cadena de codificación sea más extensa para bytes que menos se repiten. Tal es así que se observa que la eficiencia puede variar según la naturaleza de la imagen ya que en la experiencia para un archivo de imagen en el infrarrojo y con tamaño inicial de 1.34Gb se obtuvo un tamaño final de 0.831Gb, este resultado fue el de mejor ratio de compresión.

En resumen, los resultados obtenidos respaldan parcialmente los objetivos e hipótesis planteados; demostrándose que la librería hibrida OpenMPI/OpenMP mejora el tiempo

Tesis publicada con autorización del autor
No olvide citar esta tesis

de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman. Todavía es necesario mejorar el algoritmo en cuanto a la eficiencia en la cadena de codificación para el caso de imágenes con menor cantidad de bytes que se repitan.

Como complemento es importante resaltar aspectos técnicos significativos que ayudaron en esta investigación:

Accesibilidad y Facilidad de Uso: la descripción de las funcionalidades resalta la accesibilidad y facilidad de uso de la librería. (véase Anexo 4)

Integración Técnica: la utilización de herramientas tecnológicas como el compilador g++, las librerías OpenMP, las librerías OpenMPI y el sistema operativo Linux Ubuntu demuestra una sólida base técnica para el desarrollo de la aplicación. Todas las herramientas son compatibles en el sistema operativo Linux.

Adaptabilidad al algoritmo estándar Huffman: la representación del paralelismo se ajusta a las funcionalidades del estándar del algoritmo de Huffman.

En síntesis, los resultados indican que la librería hibrida OpenMPI/OpenMP cumple parcialmente con los objetivos establecidos para mejorar el procesamiento de imágenes a través de la compresión mediante el algoritmo de Huffman. Es sencillo de implementar y manipular.

Comparativa con estudios previos

Muchas de las investigaciones mencionadas han utilizado las interfaces OpenMPI y MPI en los cuales se trabaja independientemente y en algunos casos de manera hibrida (Hadjidoukas et. al, 2020). Dicha investigación da un enfoque de como implementar las dos interfaces en conjunto para el entrenamiento de redes neurales. Pero las investigaciones están enfocadas a sistemas con grandes recursos como en el caso de clúster de computadoras, sin embargo, estos recursos no son disponibles por ejemplo en los sistemas embebidos donde es necesario optimizar los procesos y las herramientas de compilación.

Una investigación interesante es también la realizada por Yin & Shi (2022) para el caso de procesamiento de gran volumen de datos heterogéneos basados en una combinación de MPI y OpenMP. En este caso el paralelismo fue orientado al procesamiento de big data y en ello se demostró que el modelo hibrido permite optimizar la comunicación paralela. En nuestra investigación se ha aplicado el mismo criterio, pero al proceso de paralelizar y optimizar el algoritmo de Huffman en la compresión de archivos de imágenes satelitales.

Limitaciones

Aunque la presente investigación se ha realizado sobre un procesador AMD de 2 núcleos, sería interesante poder desarrollar esta experiencia sobre un procesador con mayor cantidad de núcleos para validar el balanceo de carga entre los núcleos también optimizar el paralelismo de la comunicación hibrida entre OpenMPI y OpenMP.

Pese a que el algoritmo de Huffman es sencillo de implementar se ha observado que no es eficiente en algunos archivos de imágenes. Esto debido a la naturaleza propia del

Tesis publicada con autorización del autor
No olvide citar esta tesis

algoritmo en donde debido a la cadena o prefijo de codificación un byte que no se repite con frecuencia tiende a representarse con un prefijo de mayor tamaño.

Aporte a la comunidad científica

El aporte a la comunidad científica está enfocado a optimizar el proceso de la compresión de datos sin perdida de información. Proponiendo una técnica hibrida para el mejor desempeño en la utilización de recursos tanto de procesador como memoria. Ello ha demostrado reducir el tiempo de procesamiento, pero aún no en el tamaño final del archivo.

Futuras Investigaciones

Con el propósito de mejorar la representación del prefijo de codificación para hacer más compacto la compresión del archivo seria interesante utilizar el algoritmo canónico de Huffman paralelizando el procedimiento. En este algoritmo que es una variante del Huffman estándar, las longitudes de los códigos se ordenan primero por la longitud del código y luego por el valor alfabético. Se asigna a cada símbolo un código binario. El primer símbolo recibe un código con la misma longitud que el original, pero compuesto solo por ceros. A los siguientes símbolos se les asigna el siguiente número binario en la secuencia, asegurándose de que los códigos sean siempre superiores en valor. Cuando se alcanza el código más largo, se añaden ceros hasta que la longitud del nuevo código sea igual a la del código original.

La ventaja de este método es que permite codificar el diccionario de códigos en menos bits que un árbol de Huffman completamente descrito. Por lo tanto, se puede generar primero un árbol de Huffman normal y luego convertirlo en un árbol de Huffman canónico antes de usarlo. De esta manera se puede describir el código de una manera más compacta.

Una investigación interesante seria también en poder aplicar estas técnicas de paralelización en los otros algoritmos de compresión de datos imágenes como el algoritmo JPEG.

10.0 CONCLUSIONES

La implementación de una librería hibrida OpenMPI/OpenMP ha demostrado ser una estrategia efectiva para mejorar el tiempo de ejecución en la compresión de imágenes satelitales. Los resultados obtenidos respaldan la eficacia y utilidad de esta herramienta. Tras analizar los resultados derivados de la investigación acerca del uso de la librería hibrida se concluye lo siguiente:

Se encontró que la dispersión de las mediciones relacionadas al tiempo de ejecución es menor cuando se ha utilizado la librería de acuerdo al análisis estadístico discutido.

En el caso del ratio de compresión con respecto al tamaño final del archivo. Pese a que se ha demostrado que el algoritmo no es muy eficiente, el algoritmo muestra un valor interesante para el caso de archivos con tamaño 1.34Gb.

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

11.0 REFERENCIAS

Abbasi, R., Bashir, A. K., Almagrabi, A. O., Heyat, M. B. B., & Yuan, G. (2023). Efficient lossless based secure communication in 6G Internet-of-Things environments. *Sustainable Energy Technologies and Assessments,* 57, 103218.

Aldinucci, M., Cesare, V., Colonnelli, I., Martinelli, A. R., Mittone, G., Cantalupo, B., ... & Drocco, M. (2021). Practical parallelization of scientific applications with OpenMP, OpenACC and MPI. *Journal of parallel and distributed computing*, 157, 13-29

Alluru, K., & Jeganathan, L. (2021). Code Optimizations for Parallelization of Programs using Data Dependence Identifier. *International Journal of Advanced Computer Science and Applications*, 12(10).

Bernal C.A. (2015). Metodología de la Investigación Científica (4.ª ed.). Pearson.

Bhandari, P. (2023). Random Assignment in experiments | Introduction & examples. Scribbr. https://www.scribbr.com/methodology/random-assignment/

Bernal F, Albarracín C, Gaona J, Giraldo L, Mosquera C, Peña S, Torres Y, Ovalle J, Nieto J, Chacón D, Salcedo S, Suarez A, Cortés D, Pinzón J, Higuera P, Baquero C. Programación Paralela Recuperado de http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teoria/index.html#twelve

Gómez, E. S. (2020). MPI vs OpenMP: A case study on parallel generation of Mandelbrot set. *Innovación y Software*, 1(2), 12-26.

Gonnord, L., Henrio, L., Morel, L., & Radanne, G. (2022). A Survey on Parallelism and Determinism. *ACM Computing Surveys* (CSUR).

Gropp W, Lusk E, Skjellum A. (2014). Using MPI. The MIT Press

Hadjidoukas, P. E., Bartezzaghi, A., Scheidegger, F., Istrate, R., Bekas, C., & Malossi, A. C. I. (2020). torcpy: Supporting task parallelism in Python. *SoftwareX*, 12, 100517.

Harel, R. E., Pinter, Y., & Oren, G. (2022). Learning to Parallelize in a Shared-Memory Environment with Transformers. *arXiv preprint arXiv*:2204.12835.

He, L., Chen, J., Lu, J., Yan, Y., Yang, J., Yuan, G., ... & Li, Q. (2022). A Hybrid MPI/OpenMP Parallelization Scheme Based on Nested FDTD for Parametric Decay Instability. *Atmosphere,* 13(3), 472.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. del P. (2014). Metodología de la Investigación (6.ª ed.). Interamericana editores, s.a. de c.v.

Li, L., Liu, H., Zhu, Y., Liang, X., & Liu, L. (2020). A Lossless Compression Algorithm Based on Differential and Canonical Huffman Encoding for Spaceborne Magnetic Data. In *Proceedings of the 2020 2nd International Conference on Image, Video and Signal Processing* (pp. 115-119).

Mahammad, F. S., & Viswanatham, V. M. (2020). Performance analysis of data compression algorithms for heterogeneous architecture through parallel approach. *The Journal of Supercomputing*, 76(4), 2275-2288.

Migallón, H., López-Granado, O., Martínez-Rach, M. O., Galiano, V., & Malumbres, M. P. (2021). Load Balancing Strategies for Slice-Based Parallel Versions of JEM Video Encoder. *Algorithms*, *14*(11), 320

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Mustafa, D. (2022). A survey of performance tuning techniques and tools for parallel applications. *IEEE Access*.

Piñán, A. (2018). *Computación paralela: cluster de alto rendimiento virtual para el incremento de velocidad de cálculo de programas* [Tesis de pregrado, Universidad de Huánuco]. Repositorio Institucional – Universidad de Huánuco.

Quinn M. (2011). Parallel programming in c with mpi and open mp. Mc Graw Hill

Rosenberg, D., Mininni, P. D., Reddy, R., & Pouquet, A. (2020). GPU parallelization of a hybrid pseudospectral geophysical turbulence framework using CUDA. *Atmosphere*, 11(2), 178.

Schmidt B., González-Domínguez J., Hundt C. (2018). Parallel Programing: Concepts and Practices. Ed. Morgan Kaufmann Publishers

Shaharim, N. A. N., Chez, T. L., Zainul, A. Z., Rizuan, N., & Noor, M. (2022). Parallelization of CCSDS Hyperspectral Image Compression Using OpenMP. *Journal of Engineering*, 18(1), 1-16.

Süß, T., Nagel, L., Vef, M. A., Brinkmann, A., Feld, D., & Soddemann, T. (2021). Pure functions in C: A small keyword for automatic parallelization. *International Journal of Parallel Programming*, 49, 1-24

Sznajdleder, P. (2012). Algoritmos a Fondo - Con implementaciones en c y java. Ed. Alfaomega.

Yin, F., & Shi, F. (2022). Heterogeneous Big Data Parallel Computing Optimization Model using MPI/OpenMP Hybrid and Sensor Networks. *ACM Transactions on Sensor Networks*.

Zhao, H., Beck, B., Fuller, A., & Peatman, E. (2020). EasyParallel: A GUI platform for parallelization of STRUCTURE and NEWHYBRIDS analyses. *PLoS One*, 15(4), e0232110

Zhao, Y., Si, N., Sun, Y., Gao, X., Tong, H., & Yuan, G. (2022). An Automatically Privacy Protection Solution for Implementing the Right to Be Forgotten in Embedded System. *IEEE Access*, *10*, 35146-35161.

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

12.0 ANEXOS

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Anexo 1: Matriz de consistencia

Problema general	Objetivo general	Hipótesis general	METODO
¿Cuál es la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman?	Determinar la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman	La librería con tecnología hibrida OpenMPI/OpenMP mejora el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman	Método: Hipotético deductivo Enfoque: Cuantitativo Tipo: Experimental Diseño experimental (preexperimental) Población:30 imágenes satelitales Muestra: el total de la población Técnica de recolección de datos: Observación Instrumento de recolección de datos : Matriz de datos Técnica de procesamiento de datos: Estadística descriptiva Análisis inferencial: Shapiro Wilk para determinar la normalidad. Prueba no paramétrica de Wilcoxon
Problema Especifico	**Objetivo Especifico**	**Hipótesis Específicos**	
¿Cuál es la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el tiempo de ejecución en el procesamiento de imágenes con la compresión de la información usando el algoritmo de Huffman? ¿Cuál es la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre el tamaño del archivo en el procesamiento de imágenes con la compresión de la información usando el algoritmo de Huffman?	Determinar la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre **el tiempo de ejecución** en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman. Determinar la influencia de la librería con tecnología hibrida OpenMPI/OpenMP sobre **el tamaño del archivo** en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman	La librería con tecnología hibrida OpenMPI/OpenMP mejora el tiempo de ejecución en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman. La librería con tecnología hibrida OpenMPI/OpenMP mejora el tamaño del archivo en el procesamiento de imágenes en la compresión de la información usando el algoritmo de Huffman	

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Anexo 2: Matriz de operacionalización

VARIABLES	DEFINICION	DIMENSION	INDICADOR	INSTRUMENTO
Variable Independiente	Librería hibrida OpenMPI/OpenMP			
Variable Dependiente	Procesamiento de imágenes	Tiempo de ejecución	Cantidad de hilos de procesamiento	Ficha de observación
			Tiempo de ejecución inicial	
			Tiempo de ejecución final	
		Tamaño del archivo	Rango de compresión	
			Tamaño del archivo inicial	
			Tamaño del archivo final	

Tesis publicada con autorización del autor
No olvide citar esta tesis

Anexo 3: Ficha de observación/instrumento de evaluación

Título de la tesis: Desarrollo de librería para procesamiento de imágenes usando OpenMPI y OpenMP						
Variable dependiente: Procesamiento y gestión de imágenes						
Dimensión	Tiempo de ejecución			Tamaño del archivo		
Indicadores	Cantidad de hilos de procesamiento	Tiempo de ejecución inicial (seg)	Tiempo de ejecución final (seg)	Tamaño de archivo inicial (bytes)	Tamaño de archivo final (bytes)	Rango de compresión
Caso por medir						
Imagen 1	2	22.54646	2.32373	32495396	32391867	0.9968
Imagen 2	2	12.813944	0.782002	8532656	8436890	0.9888
Imagen 3	2	23.00037	3.40157	46344368	46223826	0.9974
Imagen 4	2	27.58025	5.6959	64209968	64128960	0.9987
Imagen 5	2	28.23148	4.80796	69417726	69418173	1
Imagen 6	2	28.42641	5.10154	65412889	65314086	0.9985
Imagen 7	2	20.33681	4.08408	57028549	56979783	0.9991
Imagen 8	2	28.15238	5.75589	79723507	79676218	0.9994
Imagen 9	2	20.57748	2.98922	45564344	45485859	0.9983
Imagen 10	2	28.81504	5.37917	74948320	74931928	0.9998
Imagen 11	2	28.43776	4.86494	65483223	65448593	0.9995
Imagen 12	2	22.47594	3.17312	42677888	42574525	0.9976
Imagen 13	2	31.8379	9.11703	132065314	132066094	1
Imagen 14	2	16.66437	5.23894	77088690	77084105	0.9999
Imagen 15	2	27.8894	5.43431	74850105	74824176	0.9997
Imagen 16	2	26.16378	6.15176	76808403	76789039	0.9997

Tesis publicada con autorización del autor
No olvide citar esta tesis

Imagen 17	2	32.3816	10.6205	133237031	133237811	1
Imagen 18	2	12.19464	1.29726	15929314	15845301	0.9947
Imagen 19	2	13.18202	1.21597	15074578	14986268	0.9941
Imagen 20	2	8.561195	0.881713	7106922	6950997	0.9781
Imagen 21	2	10.19444	1.23597	15218765	15131122	0.9942
Imagen 22	2	13.10668	1.20977	15133264	15042775	0.994
Imagen 23	2	11.1758	1.15287	13405310	13307091	0.9927
Imagen 24	2	12.115594	1.17121	15253636	15164151	0.9941
Imagen 25	2	11.9853	1.04457	11342815	11233522	0.9904
Imagen 26	2	19.28729	1.35205	15591890	15505059	0.9944
Imagen 27	2	18.50507	1.30291	15728469	15640996	0.9944
Imagen 28	2	23.86164	5.68984	91094944	80247944	0.8809
Imagen 29	2	69.4924	39.3206	482307868	366656648	0.7602
Imagen 30	2	280.2378	206.279	1446813024	1385422858	0.9921

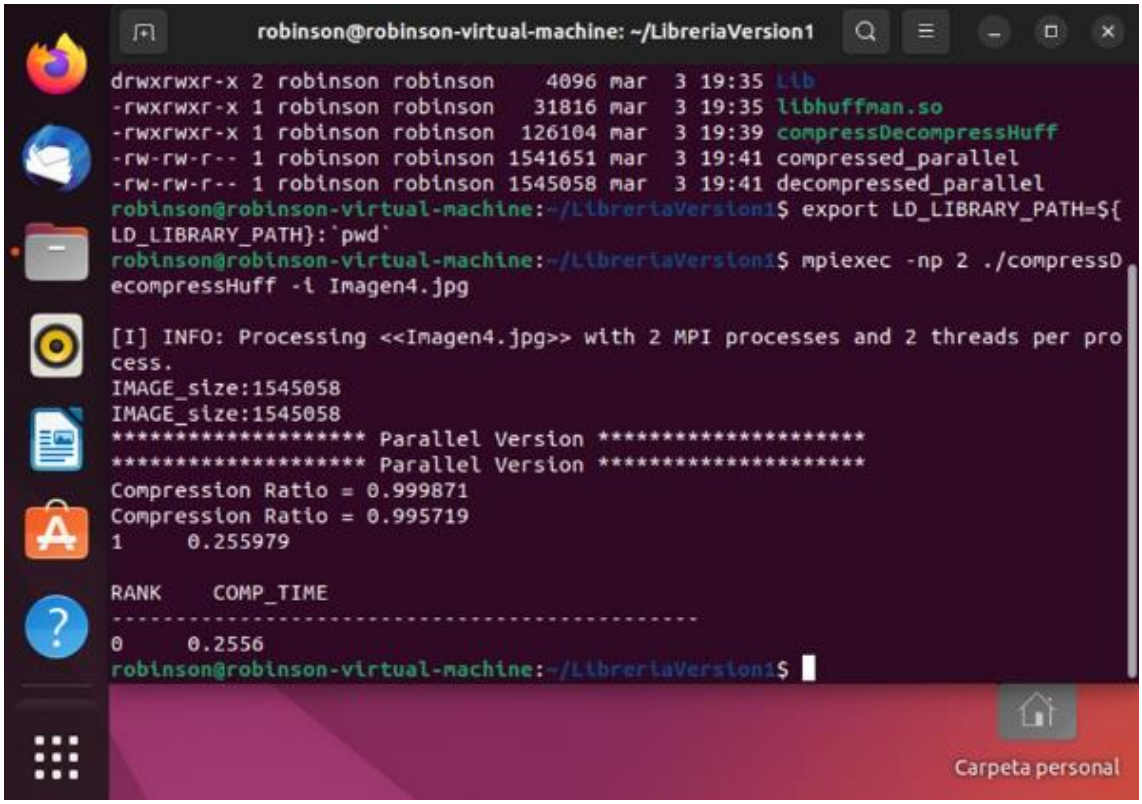
Imagen 1	**MOD09GA.A2024096.h00v08.061.2024098033105.hdf**
Imagen 2	**MOD09GA.A2024096.h00v10.061.2024098033330.hdf**
Imagen 3	**MOD09GA.A2024096.h01v07.061.2024098033652.hdf**
Imagen 4	**MOD09GA.A2024096.h02v08.061.2024098033251.hdf**
Imagen 5	**MOD09GA.A2024096.h02v10.061.2024098033616.hdf**
Imagen 6	**MOD09GA.A2024096.h07v06.061.2024098033336.hdf**
Imagen 7	**MOD09GA.A2024096.h10v11.061.2024098033544.hdf**
Imagen 8	**MOD09GA.A2024096.h11v05.061.2024098033801.hdf**
Imagen 9	**MOD09GA.A2024096.h12v10.061.2024098033544.hdf**
Imagen 10	**MOD09GA.A2024096.h13v12.061.2024098033545.hdf**
Imagen 11	**MOD09GA.A2024096.h14v11.061.2024098034734.hdf**
Imagen 12	**MOD09GA.A2024096.h15v07.061.2024098033303.hdf**
Imagen 13	**MOD09GA.A2024096.h20v03.061.2024098034009.hdf**
Imagen 14	**MOD09GA.A2024096.h21v06.061.2024098033636.hdf**
Imagen 15	**MOD09GA.A2024096.h22v05.061.2024098033821.hdf**
Imagen 16	**MOD09GA.A2024096.h26v08.061.2024098033149.hdf**
Imagen 17	**MOD09GA.A2024096.h27v03.061.2024098034319.hdf**
Imagen 18	**OMI-Aura_L2-OMMYDCLD_2024m0228t1649-o104378_v003-2024m0309t125011.he5**
Imagen 19	**OMI-Aura_L2-OMMYDCLD_2024m0228t2006-o104380_v003-2024m0309t124811.he5**
Imagen 20	**OMI-Aura_L2-OMMYDCLD_2024m0228t2324-o104382_v003-2024m0309t124238.he5**
Imagen 21	**OMI-Aura_L2-OMMYDCLD_2024m0229t0242-o104384_v003-2024m0309t124907.he5**

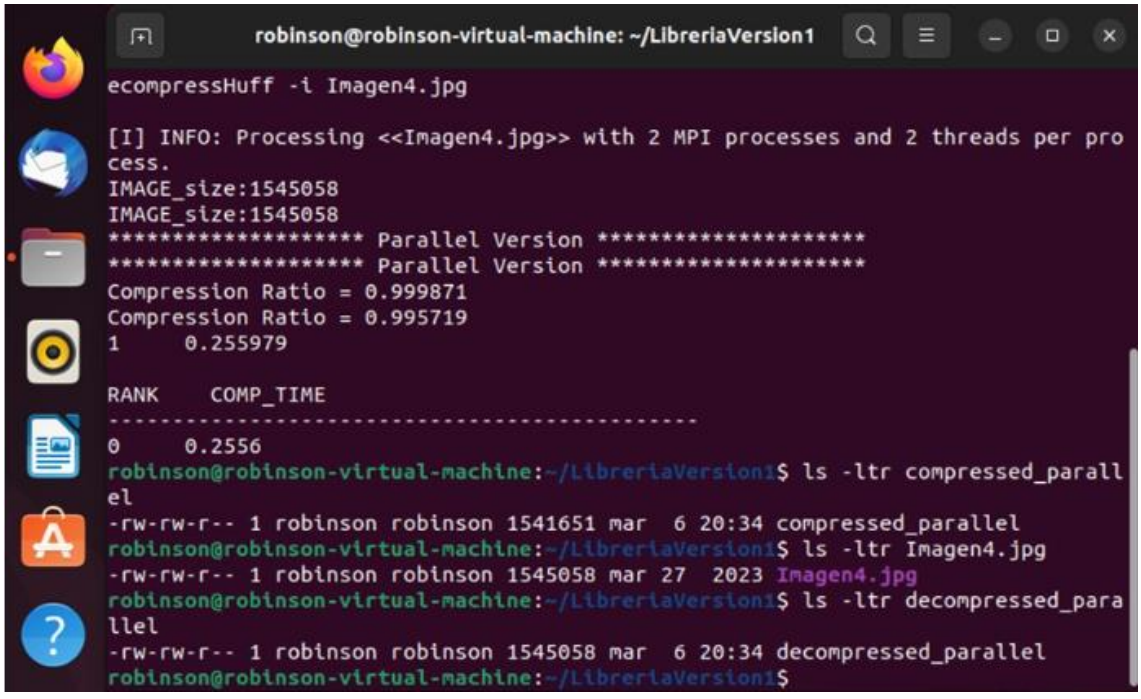
Tesis publicada con autorización del autor
No olvide citar esta tesis

Imagen 22	**OMI-Aura_L2-OMMYDCLD_2024m0229t0559-o104386_v003-2024m0309t124828.he5**
Imagen 23	**OMI-Aura_L2-OMMYDCLD_2024m0229t1235-o104390_v003-2024m0309t124733.he5**
Imagen 24	**OMI-Aura_L2-OMMYDCLD_2024m0229t1552-o104392_v003-2024m0309t124832.he5**
Imagen 25	**OMI-Aura_L2-OMMYDCLD_2024m0229t1910-o104394_v003-2024m0309t124541.he5**
Imagen 26	**OMI-Aura_L2-OMMYDCLD_2024m0229t2049-o104395_v003-2024m0309t124940.he5**
Imagen 27	**OMI-Aura_L2-OMMYDCLD_2024m0229t2228-o104396_v003-2024m0309t124942.he5**
Imagen 28	**real_final_final.tif**
Imagen 29	**real_final.tif**
Imagen 30	**infrarojo.tif**

Tesis publicada con autorización del autor
No olvide citar esta tesis

Anexo 4: Programa haciendo uso de la librería hibrida

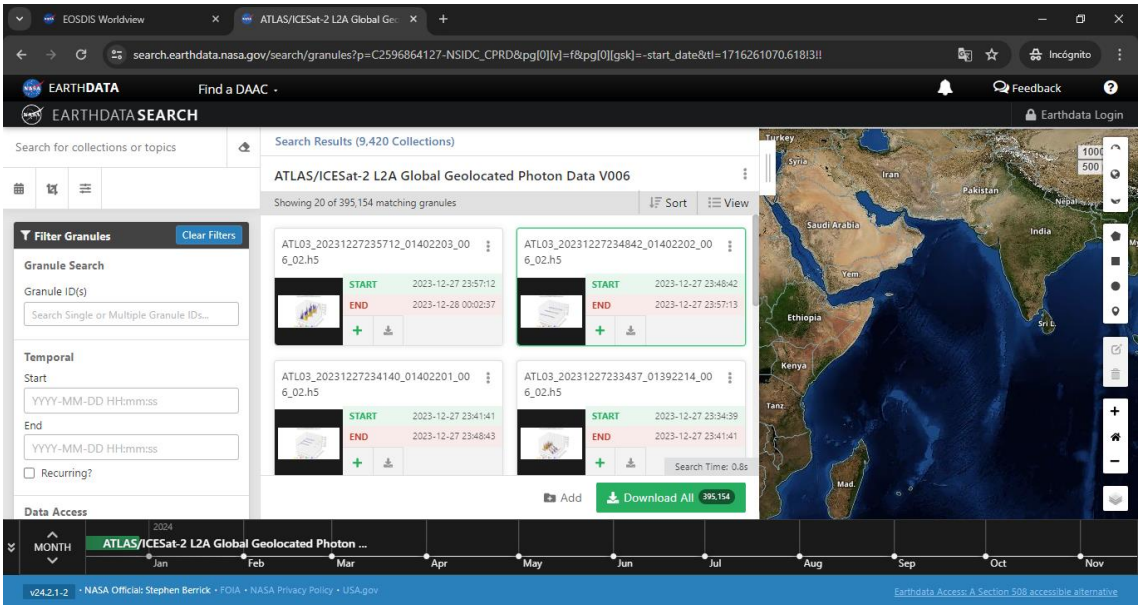


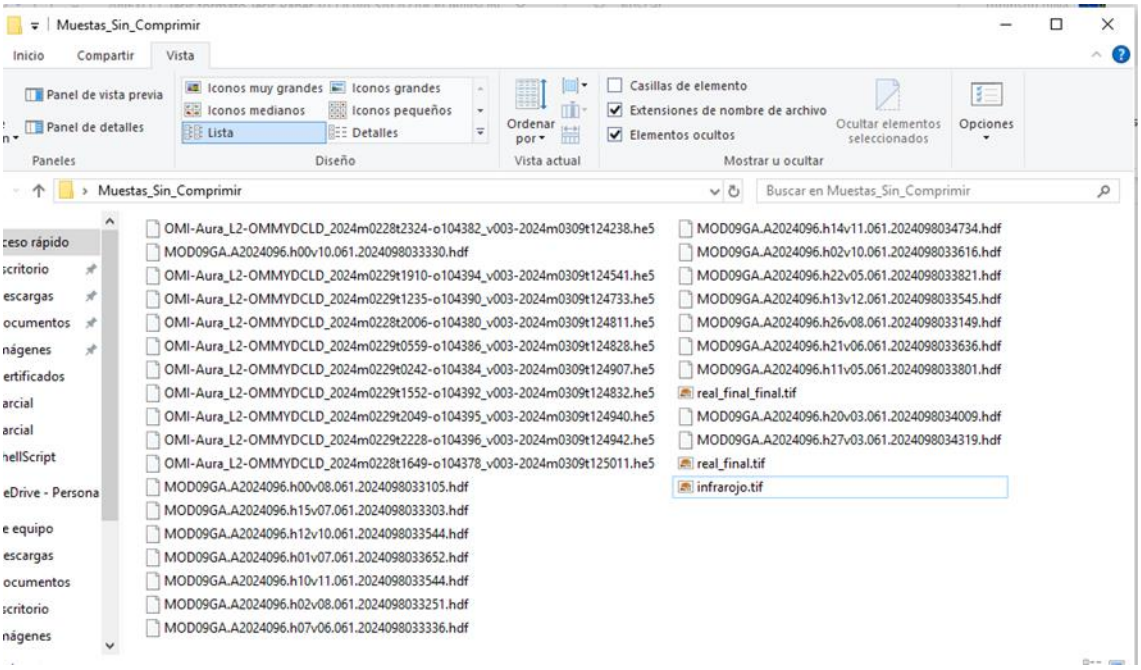


Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Anexo 5: Sistema de Información y Datos del Sistema de Observación Terrestre





Tesis publicada con autorización del autor
No olvide citar esta tesis

Anexo 6: Evidencias de envío de articulo a congreso para su publicación en revista científica



De: AUCCAHUASI AIQUIPA, WILVER <wauccahuasi@cientifica.edu.pe>
Enviado: lunes, 18 de marzo de 2024 19:57
Para: info.icoeca@gmail.com <info.icoeca@gmail.com>
Asunto: RE: I send article 3 for congress ICOECA 2024

Good morning, greetings from Lima, Peru, this is to send you article, to see the possibility of being included in the congress ICOECA 2024.

Title: Development of a library for image processing using openmpi and openmp

...

Waiting for your comments.

Best regards.

Wilver Auccahuasi

Tesis publicada con autorización del autor
No olvide citar esta tesis

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Anexo 7: Evidencias de aceptación de articulo en congreso para su publicación en revista científica

4th International Conference on Expert Clouds and Applications (ICOECA 2024)

18-19, April 2024 | Bengaluru, India

http://icoeca.com/
info.icoeca@gmail.com

Letter of Acceptance

Details of accepted paper:

Paper ID	Title	Author(s)
ICOECA-274	Development of a library for image processing using openmpi and openmp	Robinson Oliva-SalazarWilver, Wilver Auccahuasi

Herewith, the conference committee of the 4th International Conference on Expert Clouds and Applications (ICOECA 2024) extends its warmest congratulations to you. We are pleased to inform you that after a rigorous peer review process, your aforementioned research paper has been accepted for presentation at the conference. Additionally, your paper has been recommended for inclusion in the ICOECA 2024 conference proceedings.

The ICOECA conference is scheduled to take place on 18-19, April 2024, at RV College of Engineering, Bengaluru, India. We encourage the active participation of highly qualified delegates like yourself to foster the exchange of innovative research ideas and insights.

We congratulate you on being successfully selected for the presentation of your research work in our esteemed conference.

Deepika

ICOECA

Prof. Deepika K
ICOECA 2024

PUBLISHED BY IEEE COMPUTER SOCIETY
CONFERENCE PUBLISHING SERVICES

Tesis publicada con autorización del autor
No olvide citar esta tesis

RESOLUCION DIRECTORIAL DE CARRERA: Ingeniería de Sistemas de Información y Gestión -CI-DAFCE-U. CIENTIFICA-2020

CARRERA DE: Ingeniería de Sistemas de Información y Gestión

Lima 24 de 08 del 2023

VISTO:

El informe de resolución independiente académico del proyecto de tesis titulado: DESARROLLO DE LIBRERÍA PARA PROCESAMIENTO Y GESTIÓN DE IMÁGENES USANDO OPENMPI Y OPENMP, LIMA 2023 presentado por:

-ROBINSON OLIVA SALAZAR //

CONSIDERADO:

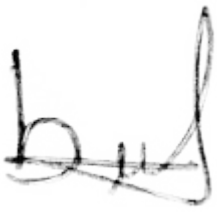
Que, de acuerdo con el reglamento general de la Universidad Científica del Sur y los reglamentos de para obtener el en la Carrera de Ingeniería de Sistemas de Información y Gestión se debe desarrollar un trabajo de investigación

Que, de acuerdo normativa vigente de la Universidad Científica del Sur, en uso de las atribuciones conferidas al director académico de carrera

SE RESUELVE:

Aprobar e inscribir el proyecto de tesis titulado: DESARROLLO DE LIBRERÍA PARA PROCESAMIENTO Y GESTIÓN DE IMÁGENES USANDO OPENMPI Y OPENMP, LIMA 2023 con Número de registro: PRE-107-2023-00218, y establecer el inicio del periodo de ejecución del mencionado proyecto.
Nombrar al docente WILVER AUCCAHUASI AIQUIPA // como asesor para el desarrollo de tesis en cuestión
Regístrese, Comuníquese y archívese

Director Académico
Carrera de Ingeniería de Sistemas de Información y Gestión

	
Dr. Sandra Jeannet Meza Balvin Director de Departamento Académico de Ingeniería de Sistemas Empresariales	**Mg. Brian Andreé Meneses Claudio** Responsable de Investigación de Ingenierías de Ciencias Empresariales

TESIS
UNIVERSIDAD CIENTÍFICA DEL SUR

Tesis publicada con autorización del autor
No olvide citar esta tesis