**Comparación de Rendimiento en Tiempo de Ejecución de los Algoritmos de Compresión de Datos Run Length y Lempel Ziv en CPU y GPU Utilizando CUDA.**

Mayta Rosas Milagros Lizet, Talavera Díaz Henry Abraham, Gonzalo E. Quispe Huanca

*Escuela Profesional de Ingeniería de Sistemas, Facultad de Ingenierías de Producción y Servicios, Universidad Nacional de San Agustín, Arequipa, Perú.*

milagros.mayta.rosas@gmail.com

hen.talavera@gmail.com

gonzqh@gmail.com

*Abstract*—**The dev………..**

*Keywords***— Machine Learning, Decision Tree, J48, Bayesian Net, Neuronal Networks, WEKA, Classification, Medical Sciences.**

*Resumen***— El desarrollo ………. factor la exactitud.**

*Palabras Clave***— Aprendizaje de Maquinas, Árbol de Decisiones, J48, Redes Bayesianas, Redes Neuronales, WEKA, Clasificación, Ciencias Médicas.**

I. INTRODUCCIÓN

GPU Computing, Uso de propósito general de una tarjeta GPU(Graphics Processing Unit), para realizar cálculos científicos.

El problema inicial del uso de las tarjetas gráficas para el cálculo científico de propósito general (GPGPU - General-Purpose Computing on Graphics Processing Units) era que se necesitaba usar lenguajes de programación específicos para gráficos como el OpenGL o el Cg para programar la GPU. Los desarrolladores debían hacer que sus aplicaciones científicas parecieran aplicaciones gráficas convirtiéndolas en problemas que dibujaban triángulos y polígonos. Esto claramente limitaba el acceso por parte del mundo científico al enorme rendimiento de las GPUs.

NVIDIA fue consciente del potencial que suponía acercar este enorme rendimiento a la comunidad científica en general y decidió investigar la forma de modificar la arquitectura de sus GPUs para que fueran completamente programables para aplicaciones científicas además de añadir soporte para lenguajes de alto nivel como C y C++. De este modo, en Noviembre de 2009, NVIDIA introdujo para sus tarjetas gráficas la arquitectura CUDA (Compute Unified Device Architecture), una nueva arquitectura para cálculo paralelo de propósito general, con un nuevo repertorio de instrucciones y un nuevo modelo de programación paralela, con soporte para lenguajes de alto nivel (C, OpenCL, C++) y constituidas por cientos de núcleos que pueden procesar de manera concurrente miles de hilos de ejecución. En esta arquitectura, cada núcleo tiene ciertos recursos compartidos, incluyendo registros y memoria. La memoria compartida integrada en el propio chip permite que las tareas que se están ejecutando en estos núcleos compartan datos sin tener que enviarlos a través del bus de memoria del sistema [1].

II. ESTADO DEL ARTE

Se han presentado diversos artículos científicos relacionados con el uso de CUDA y la paralelización de algoritmos secuenciales.

Patel, Zhang, Mak, Davidson y Owens presentan algunos algoritmos paralelos e implementaciones de un esquema de compresión de datos sin pérdidas tipo bzip2 para arquitecturas de GPU, su enfoque paraleliza tres etapas principales en la tubería de compresión bzip2: transformación de Burrows-Wheeler (BWT), transformación de movimiento a frente (MTF) y codificación de Huffman [3].

En [5], los autores presentan la implementación de un método de compresión de datos de imágenes espectrales llamado Linear Prediction con Coeficientes Constantes (LP-CC) usando la arquitectura CUDA de computación paralela de Nvidia, su implementación de la GPU se compara experimentalmente con la implementación nativa de la CPU.

En [6], los autores exploran las posibles mejoras de rendimiento que podrían obtenerse mediante el uso de técnicas de procesamiento de GPU dentro de la arquitectura CUDA para el algoritmo de compresión JPEG. La elección de algoritmos de compresión como el foco se basó en ejemplos de paralelismo de nivel de datos que se encuentran dentro de los algoritmos y un deseo de explorar la eficacia de la gestión de algoritmos cooperativos entre el sistema de CPU y una GPU disponible.

Cloud, Curry, Ward, SKjellum y Bangalore, en [7], presentan una modificación del algoritmo de Huffman que permite que los datos sin comprimir se descompongan en bloques independientes compresibles y descomprimibles, permitiendo la compresión y descompresión concurrentes en múltiples procesadores, modificado en una GPU NVIDIA,mostrando un rendimiento favorable de GPU para casi todas las pruebas.

En [8], los autores implementan nueve esquemas de compresión ligeros en la GPU y estudian las combinaciones de estos esquemas para una mejor relación de compresión. Diseñan un planificador de compresión para encontrar la combinación óptima y sus experimentos demuestran que la compresión basada en GPU y la descompresión alcanzaron una velocidad de procesamiento de hasta 45 y 56 GB / s, respectivamente.

En la investigación de Franco, Bernabé y Acacio en [9] nos presentan la paralelización en CUDA de una transformada wavelet en 2D en una tarjeta gráfica la NVIDIA Tesla C870, con la cual, logran alcanzar una aceleración de 20.8 para un tamaño de 8192 x 8192 en comparación con la implementación en OpenMP.

REFERENCIAS

1. C. Represa, J. Cámara, P. Sánchez, “Introducción a la Programación en CUDA” Universidad de Burgos.
2. A. Ozsoy, M.Swany, “CULZSS: LZSS lossless data compression on CUDA” Universidad de Delaware
3. R. Pathel, Y.Zhang, “Parallel Lossless Data Compression on the GPU” Universidad de California
4. J. Gilchrist, “Parallel Data Compression with BZIP2” Universidad de Carleton
5. J. Mielikainen “GPUs for data parallel spectral image compression” Proceedings of SPIE
6. P. Patel, J. Wong, M. Tatikonda, J. Marczewski “JPEG Compression Algorithm Using CUDA” Universidad de Toronto
7. R.L. Cloud, M.L. Curry “Accelerating Lossless Data Compression with GPUs”
8. Q. Luo “Database Compression on Graphics Processors ” Universidad de Hong Kong
9. J. Franco, G.Bernabe “A Parallel Implementation of the 2D Wavelet Transform Using CUDA” Universidad de Murcia