CÓMPUTO PARALELO EN UNIDADES DE PROCESAMIENTO GRÁFICO Y MEDICIÓN DE TIEMPOS DE RESPUESTA COMO CRITERIO DE DESEMPEÑO

PARALLEL COMPUTING IN GRAPHIC PROCESSING UNITS AND MEASUREMENT OF RESPONSE TIMES AS PERFORMANCE CRITERIA

David Mendieta Morales
Instituto Politécnico Nacional, México
Davidmm06@outlook.com

Diana Lizet González Baldovinos
Instituto Politécnico Nacional, México
glez_lizet@hotmail.com

Jose Luis Cano Rosas
Instituto Politécnico Nacional, México
lucskyr@gmail.com

Pedro Guevara López
Instituto Politécnico Nacional, México
pguevara@real-time.com.mx

Recepción: 30/octubre/2020 **Aceptación:** 10/diciembre/2020

Resumen

En este trabajo se presenta el procedimiento para la paralelización de procesos en una Unidad de Procesamiento Gráfico (GPU) y la medición de sus tiempos de respuesta como criterio de desempeño. Para el desarrollo de este trabajo se realizaron pruebas experimentales en dos bancos de prueba: el primero está conformado por una GPU NVIDIA GTX 1050 adaptada de forma externa a una laptop; el segundo, una laptop con GPU integrada NVIDIA GTX 1050. Cada banco de pruebas está configurado para dos sistemas operativos, Windows 10 y GNU/Linux Ubuntu con la plataforma CUDA. Como caso de estudio se utilizan dos algoritmos complejos, producto de matrices y eliminación de matrices por método de Gauss-Jordan, donde se realiza un conjunto de mediciones de tiempos de respuesta, analizando su dinámica y desempeño a través de sus primeros momentos de probabilidad (media y varianza).

Palabras Clave: Algoritmo complejo, CUDA, desempeño, GPU, paralelización de procesos.

Abstract

This paper presents the parallelization of processes in a Graphic Processing Unit (GPU) and measurement of their response times as performance criteria. To support this work, experimental tests have been carried out on two test benches; The first one is made up of an NVIDIA GTX 1050 Graphics Processing Unit externally adapted to a laptop; The second one is a laptop with an integrated GPU. Each test bench is configured for two operating systems, Windows 10 and GNU/Linux Ubuntu with the CUDA platform. As a test object, two complex algorithms are used, product of matrices and elimination of matrices by the Gauss-Jordan method, where a set of response time measurements is made, analyzing their dynamics and performance through their first moments of probability (mean and variance).

***Keywords*:** *Complex algorithm, CUDA, GPU, parallelization of processes, performance.*

1. Introducción

En la actualidad, la computación paralela se utiliza en una multitud de campos para el desarrollo de aplicaciones y el estudio de problemas que requieren de un alto poder de procesamiento, ya sea por la complejidad de los problemas que abordan o por la necesidad de trabajar con problemas en tiempo real. Entre los campos que se destacan se encuentran : previsión meteorológica numérica, oceanografía, investigación médica y desarrollo industrial].

Por ello, surge la necesidad del uso de computadoras de alto desempeño, estaciones de trabajo o supercómputo, sin embargo, muchas veces es complicado tener acceso a este tipo de equipos, desaprovechando el verdadero potencial de computadoras de escritorio y laptops comunes que disponen de tarjetas de video con GPU's NVIDIA capaces de llevar a cabo procesamiento paralelo.De acuerdo con [Soto, 2016] la computación paralela sobre GPU es un modelo de computación heterogéneo, ya que este consiste en el uso conjunto de una CPU y una GPU, en

el cual, la parte secuencial del código se ejecuta sobre la CPU (host) y la parte donde se realizarán todos los cálculos se ejecutara sobre la GPU (device), el problema principal que presentaba el uso de tarjetas gráficas en aplicaciones científicas era que se necesitaba convertir la aplicación de forma gráfica usando lenguajes específicos como OpenGL o CG convirtiendo los problemas en figuras como triángulos y polígonos, desaprovechando el potencial de cálculo de una tarjeta gráfica, por esta razón, NVIDIA introdujo a sus tarjetas gráficas la arquitectura CUDA (Arquitectura Unificada de dispositivos de Cómputo), en ésta se incluyó soporte para lenguajes de alto nivel como C y C++.

Existen trabajos estrechamente relacionados, dentro de los cuales se han tomado los de mayor relevancia:

- En [Molina, 2012], los autores muestran a través de ejemplos el comportamiento que presentan los algoritmos paralelos y la eficiencia obtenida al implementarlos sobre sistemas multinúcleo; el trabajo no aborda con profundidad conceptos relacionados con el tema, sino que presenta un análisis de rendimiento para destacar la diferencia en tiempos de ejecución de algoritmos paralelos respecto a los secuenciales, mostrando al lector su indiscutible potencial.
- En [Ccapacca, 2012], Paralelización sobre GPU del algoritmo de eliminación de Gauss-Jordan para la solución de sistemas de ecuaciones lineales, se realiza la construcción de dos modelos de implementación para el algoritmo de eliminación de Gauss-Jordan para la solución de sistemas de ecuaciones lineales de la forma $AX = B$, con la finalidad de acelerar el proceso de reducción de la matriz. En [Koo, 2016], se presenta el resultado del desarrollo de aplicaciones paralelas con Python, C++, OpenMP y Boost, se analiza a partir de la medición y cálculo de tres indicadores: tiempo de ejecución, aceleración y eficiencia sobre procesadores multinúcleo.
- Finalmente, en [Baldovinos, 2018a], [Baldovinos, 2018b] y [Baldovinos, 2019] se realizan interesantes análisis sobre tiempos de respuesta, donde se pueden considerar como métrica el cálculo de los primeros momentos de probabilidad (media y varianza recursiva), además de que utilizan como

banco de prueba computadoras de placa reducida y sistema operativo RT-Linux.

2. Métodos

Este trabajo se desarrolla en varias etapas, las cuales se representan a través de un estado del autómata finito presentado en la figura 1, mientras que en la tabla 1, se especifican los elementos que integran el autómata:

- **Etapa A: Inicio**
- **Etapa B: Selección de hardware para los bancos de prueba**
  - ✓ **Banco de pruebas 1**. Para este banco de pruebas, se utiliza el equipo cómputo mostrado en la figura 2 con una GPU externa. En la tabla 2 se muestran sus características.

    Hardware necesario para conectar GPU externa al equipo de cómputo:
    - Tarjeta gráfica NVIDIA Gigabyte GeForce GTX 1050 OC 2G, figura 3.
    - Adaptador Exp GDC Beast, figura 4.
    - Cable ATX a EXP GDC/ATX PSU 20 Pin + CPU 4 Pin a EXP GDC 8 Pin, figura 5.
    - Cable HDMI A NGFF M.2, figura 6.
    - Fuente de poder corsair CX650M, figura 7.
    - Cable PCI-E de 8 pines a 6 + 2 pines, figura 8.

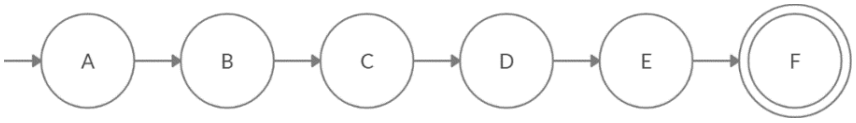

Figura 1 Autómata finito determinista de las etapas del proyecto.

Tabla 1 Elementos que integran el autómata finito determinista.

Estados	$Q = \{A, B, C, D, E, F\}$
Alfabeto	$\sum = \{\varepsilon\}$
Gramática	$G = \{\{A, B, C, D, E, F\}; \{\varepsilon\}; A; F\}$
Transiciones δ	$\delta(A, \varepsilon) = B$ $\delta(B, \varepsilon) = C$ $\delta(C, \varepsilon) = D$ $\delta(D, \varepsilon) = E$ $\delta(E, \varepsilon) = F$

Figura 2 Laptop Lenovo z40-70 utilizada como banco de pruebas 1.

Tabla 2 Características del Banco de pruebas 1.

Marca	**Lenovo**
Modelo	z40-70
CPU	Core i3-4005U

Figura 3 Tarjeta gráfica [Gigabyte, 2020].

Figura 4 Adaptador [G. Store, 2016]

Figura 5 Cable ATX a EXP [G. Store, 2016].

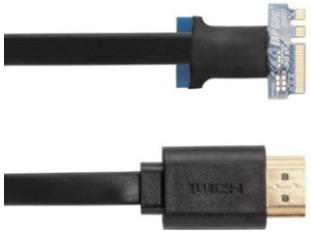

Figura 6 Cable HDMI A NGFF [G. Store, 2016].

Figura 7 Fuente de Poder [Corsair, 2015].

Figura 8 Cable PCI-E [Mercado Libre, 2020].

- ✓ **Banco de pruebas 2**. Para este banco de pruebas, se utiliza el equipo cómputo mostrado en la figura 9. En la tabla 3 se muestran sus características.

Figura 9 Laptop HP Pavilion Gaming 15-dk0xxx utilizada como banco de pruebas 2.

Tabla 3 Características de la laptop HP Pavilion Gaming Laptop 15-dk0xxx.

Marca	HP
Modelo	HP Pavilion Gaming Laptop 15-dk0xxx
Tarjeta gráfica	NVIDIA GEFORCE GTX 1050
CPU	Core i5-9300H

- **Etapa C: Montaje e integración del hardware para la adaptación del banco de pruebas 1**

  Se procede con el montaje del adaptador EXP GDC, la comunicación del adaptador con la laptop requiere un puerto NGFF M.2 que se encuentre libre para conectar el cable HDMI A NGFF M.2, en caso de no contar con un puerto libre se procede a remplazar la tarjeta de red, ya que esta se encuentra utilizando uno de estos puertos. El montaje del adaptador no lleva un orden fijo, por lo que se realiza de la siguiente manera, se conecta la tarjeta gráfica al adaptador y a este se le conecta la punta HDMI del cable HDMI A NGFF M.2, para alimentar el adaptador se utiliza el cable ATX a EXP GDC/ATX PSU 20 Pin + CPU 4 Pin a EXP GDC 8Pin a este se le conectara la alimentación principal y la alimentación del CPU de la fuente de alimentación, para alimentación de la tarjeta gráfica se le conecta el cable PCI-E de 8 pines a 6 + 2 pines de la fuente de alimentación, figura 10.

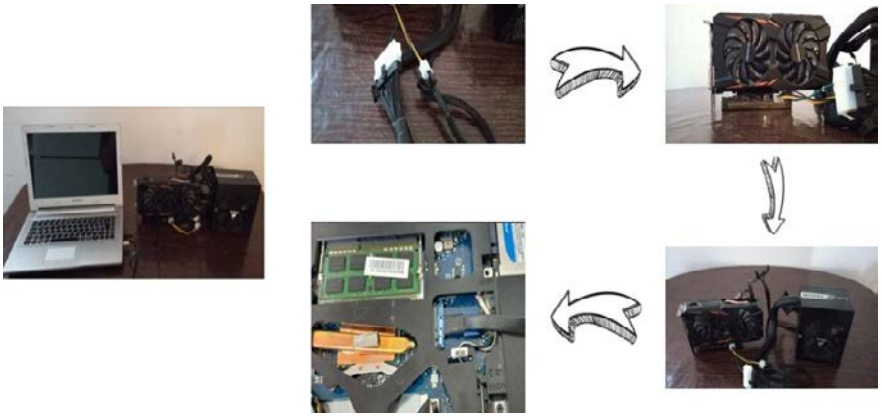

Figura 10 Montaje del adaptador EXP GDC.

- **Etapa D: Instalación del software requerido para trabajar con CUDA**

  La instalación de los drivers cambia dependiendo del sistema operativo, ya que este trabajo se realiza sobre los sistemas operativos Windows 10 y GNU/Linux Ubuntu 18.04.

  La instalación del **toolkit de CUDA** es diferente dependiendo del sistema operativo que se esté utilizando. La tabla 4 muestra la forma de instalación sobre los sistemas operativos Windows 10 y GNU/Linux Ubuntu 18.04.

Tabla 4 Metodología de instalación del Toolkit de CUDA.

Windows 10	GNU/Linux Ubuntu 18.04
Ingresar a la página: https://developer.nvidia.com/cuda- toolkit	sudo apt update && apt dist-upgrade -y && reboot
Seleccionar sistema operativo Windows	lspci -v
Seleccionar las especificaciones del equipo	sudo apt install -y ocl-icd-libopencl1 nvidia-driver-440 nvidia-cuda-toolkit
Descargar y ejecutar el toolkit CUDA versión 10.2	nvidia-smi
	sudo reboot

- **Etapa E: Programación de los algoritmos seleccionados**
  - ✓ **Algoritmo de eliminación por Gauss-Jordan**. En la programación de este algoritmo tomado de [Ccaccapa, 2012], se utiliza la memoria compartida de la tarjeta gráfica para almacenar los resultados de las operaciones realizadas, lo que agiliza el proceso de ejecución de operaciones, este proceso se conoce como reducción paralela. En la figura 11, se muestra el diagrama de flujo de este algoritmo.

- ✓ **Algoritmo de Producto de matrices**. Se seleccionó producto de matrices como objeto de prueba y se paralelizó, este algoritmo no utiliza memoria compartida, en cambio, utiliza memoria global de la tarjeta gráfica. En la figura 12, se muestra su diagrama de flujo.

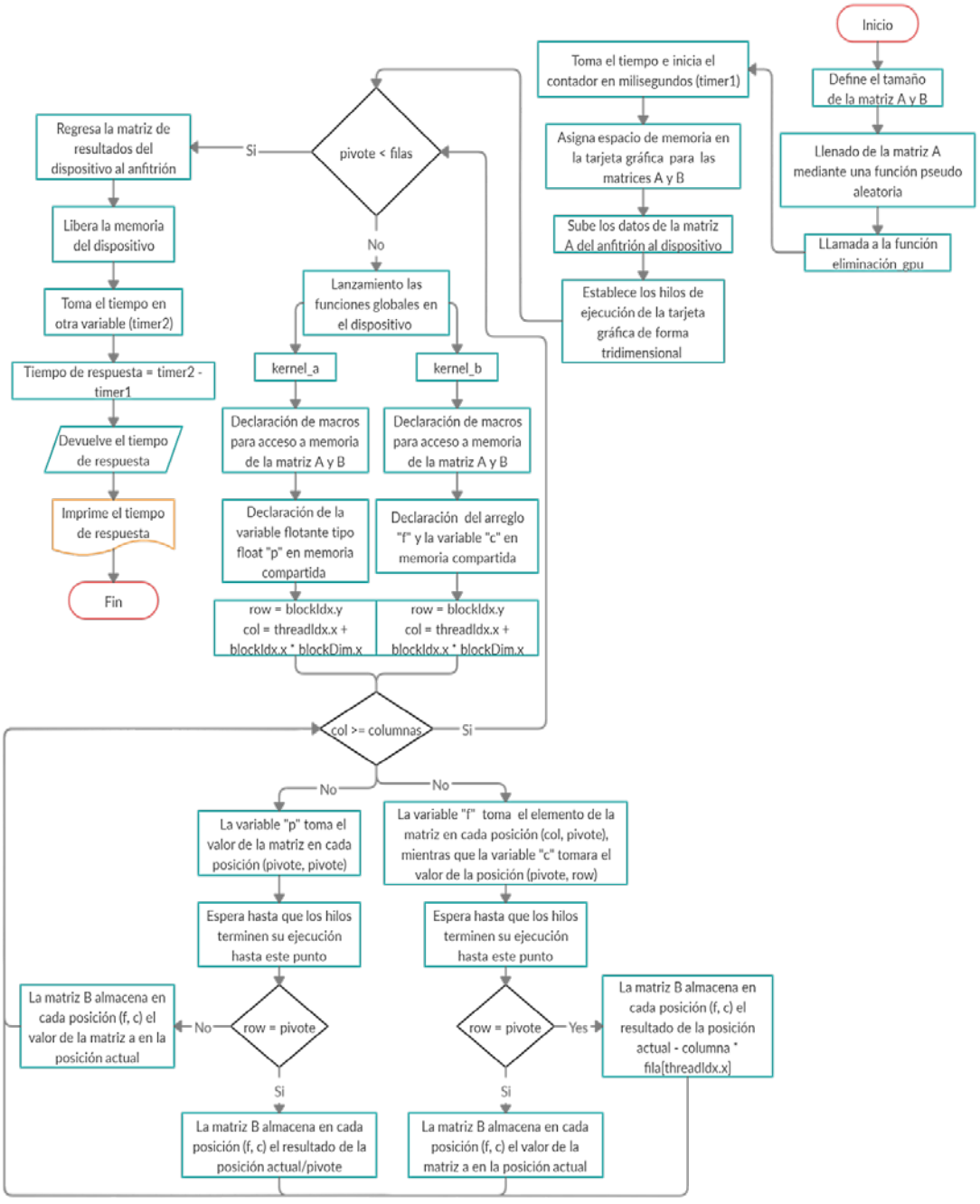

Figura 11 Diagrama de flujo del algoritmo de eliminación por Gauss-Jordan.

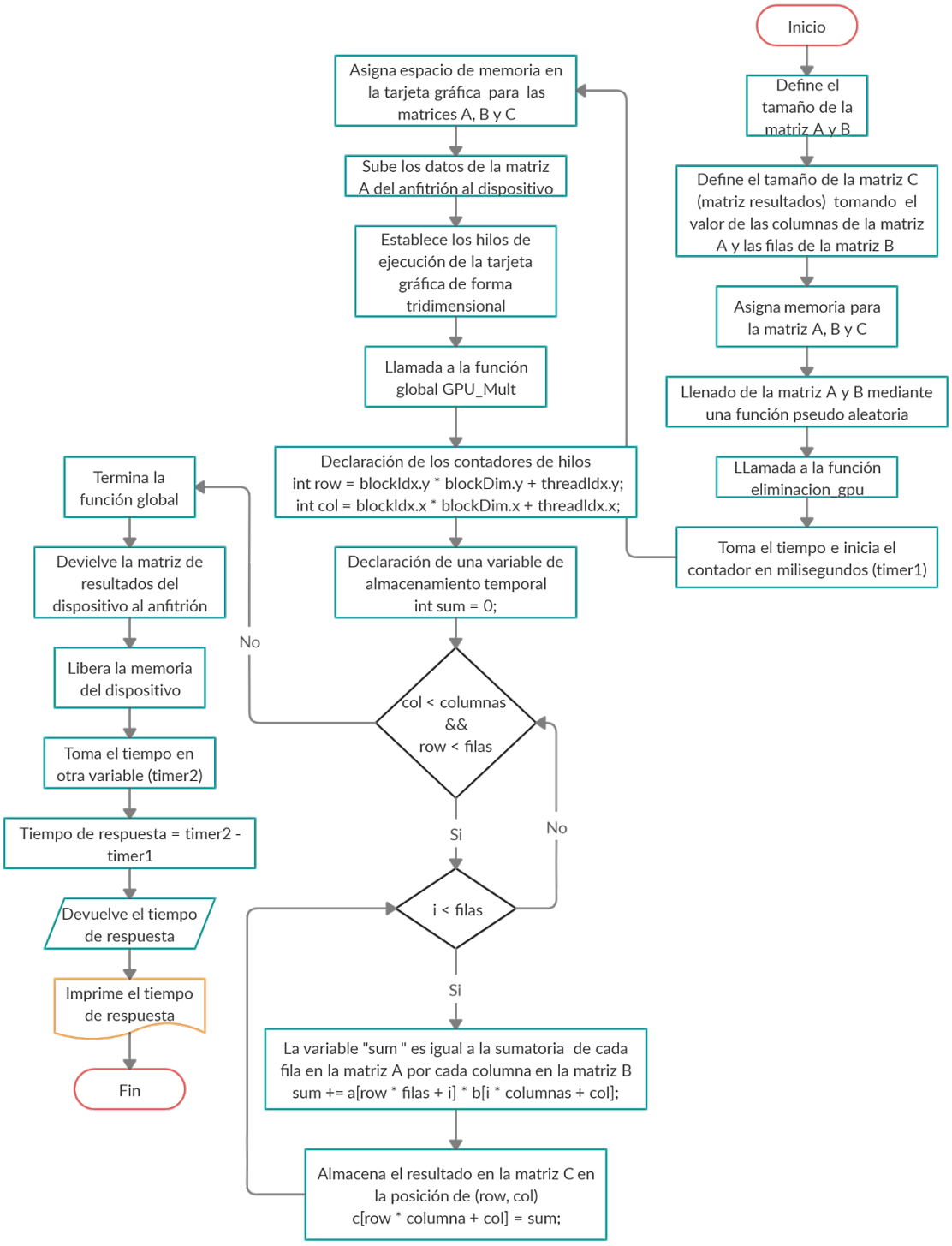

Figura 12 Diagrama de flujo del algoritmo producto de matrices.

- **Etapa F: Medición y análisis de los tiempos de respuesta**

  Para la medición los tiempos de respuesta, se utiliza una función incluida en paquete de herramientas de CUDA, esta función es cudaEventElapsedTime,

la cual calcula el tiempo transcurrido entre dos eventos en milisegundos con una resolución de alrededor de 0.5 microsegundos. Esta función utiliza dos variables tipo cudaEvent_t declaradas respectivamente como timer1y timer2, estas variables crean un objeto de evento para el dispositivo actual [NVIDIA, 2020]. Como se observa en los diagramas de flujo de las figuras 11 y 12, la medición de los tiempos de respuesta comienza desde que se reservan los recursos para la tarjeta gráfica y termina al regresar el resultado de las operaciones realizadas en la función global.

El análisis estadístico tiene un papel de especial importancia en este trabajo, ya que permite conocer la dinámica de los tiempos de respuesta en cada experimento. Este análisis se realiza fuera de línea, es decir, con los vectores obtenidos en cada experimento, se procede a realizar en el software de Matlab el cálculo de los primeros momentos de probabilidad, es decir, media y varianza [Baldovinos, 2018a].

Para realizar el análisis estadístico de los tiempos de respuesta, se toman como referencia los trabajos [Baldovinos, 2018a], [Baldovinos, 2018b] y [Baldovinos, 2019] en los que se toma el primer momento de probabilidad como la *media o valor esperado*, denotada en la ecuación 1, en este caso la variable R_k representa el tiempo de respuesta medido.

$$\mu(R_k) = \frac{1}{k}\sum_{k=1}^{k} R_k \tag{1}$$

Sin embargo, es importante observar el valor medio del tiempo de respuesta a medida que evoluciona el algoritmo, por lo tanto, es necesario tener la ecuación de la media recursiva, la cual se representa en la ecuación 2, con el objeto de conocer el comportamiento global y observar la dinámica de los tiempos de respuesta [Baldovinos, 2018a] [Baldovinos, 2018b].

$$\mu(R_k) = \frac{(k-1)\mu(R_{k-1}) + R_k}{k} \tag{2}$$

El segundo momento de probabilidad o *varianza* del tiempo de respuesta R_k es una medida de la dispersión de sus valores alrededor de la media μ y se

denota por σ^2. La expresión matemática de la varianza se presenta en la ecuación 3 [Baldovinos, 2018a] [Baldovinos, 2018b].

$$\sigma^2(R_k) = \frac{1}{k}\sum_{K=1}^{k} (R_k - \mu(R_k))^2 \quad (3)$$

De igual forma que el primer momento de probabilidad, la ecuación 3 entrega el valor de varianza de los tiempos de respuesta del número total de iteraciones. No obstante, es necesario analizar la varianza de los tiempos de respuesta a medida que evoluciona el algoritmo. Para lograr realizar este análisis, se necesita obtener la ecuación de la varianza recursiva, para ello se parte de la ecuación original 3, y se aplica diferencias finitas para obtener la varianza recursiva, representada por la ecuación 4 [Baldovinos, 2018a] [Baldovinos, 2018b].

$$\sigma^2(R_k) = \frac{(k-1)\sigma^2(R_{k-1}) + \left(R_k - \mu(R_k)\right)^2}{k} \quad (4)$$

3. Resultados

En esta sección, se muestra una colección de gráficas (figuras 13, 14, 15 y 16) de los resultados experimentales, a su vez, se encuentra dividida en dos partes:

- La primera (figuras 13 y 14), corresponde a los resultados experimentales obtenidos en el banco de pruebas 1 (GPU externa).
- La segunda (figuras 15 y 16), pertenece a los resultados obtenidos en el banco de pruebas dos (GPU integrada).

Cada una con sus respectivas gráficas de tiempos de respuesta generados por la ejecución de los algoritmos, producto de matrices y eliminación de matrices por Gauss-Jordan, ejecutados en los sistemas operativos Windows 10 y GNU/Linux Ubuntu 18.04, mostrando su primer y segundo momentos de probabilidad. Cabe resaltar que, de todo el conjunto de experimentos realizados, para este trabajo se han seleccionado las pruebas más representativas con matriz de dimensión 3000×3000 con 200 eventos.

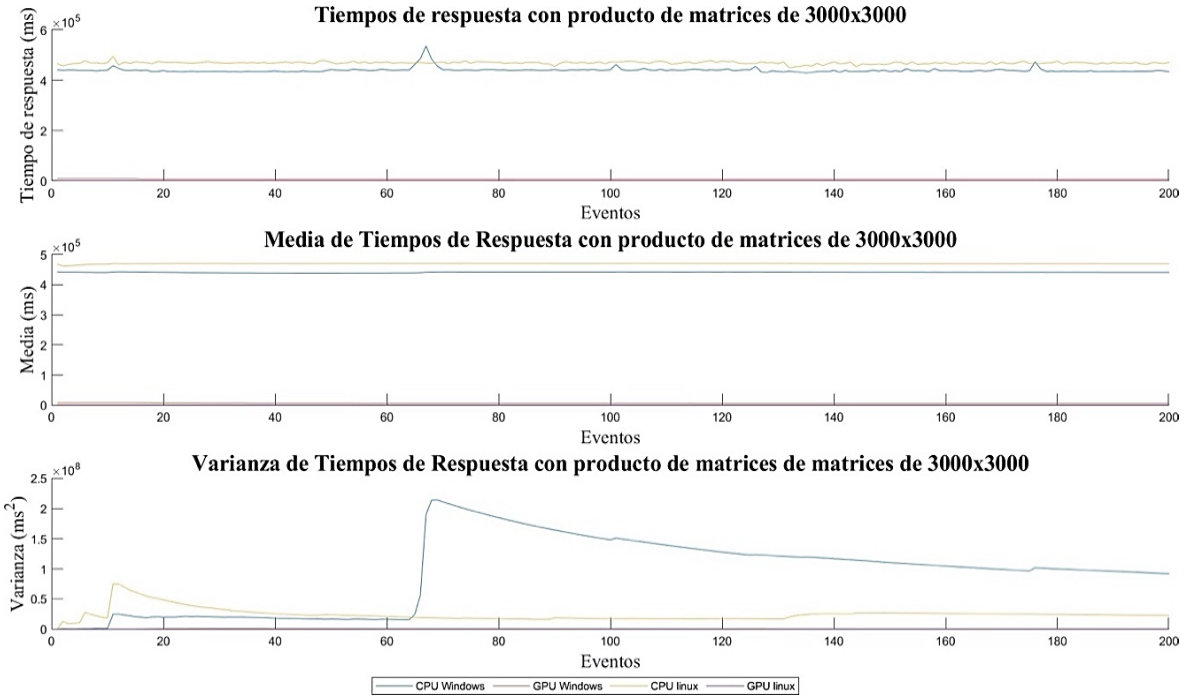

Figura 13 Tiempos de respuesta con producto de matrices de 3000×3000 (GPU externa).

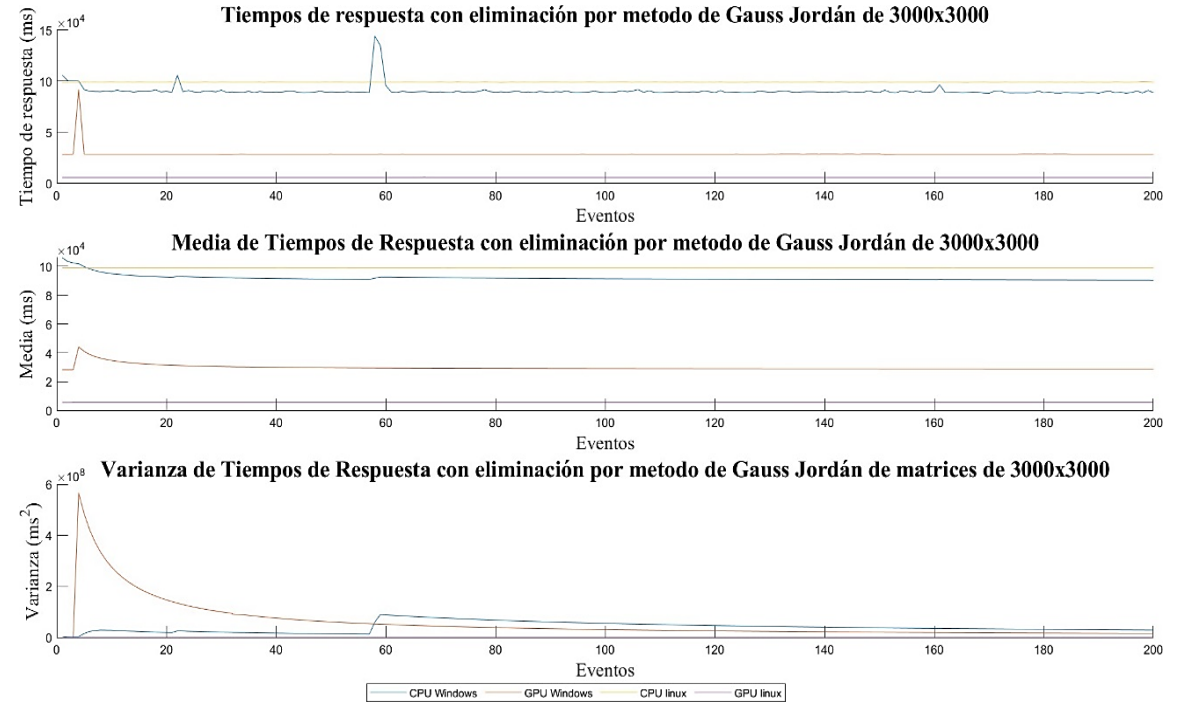

Figura 14 Tiempos de respuesta con eliminación por método de Gauss-Jordan de matrices de 3000×3000 (GPU externa).

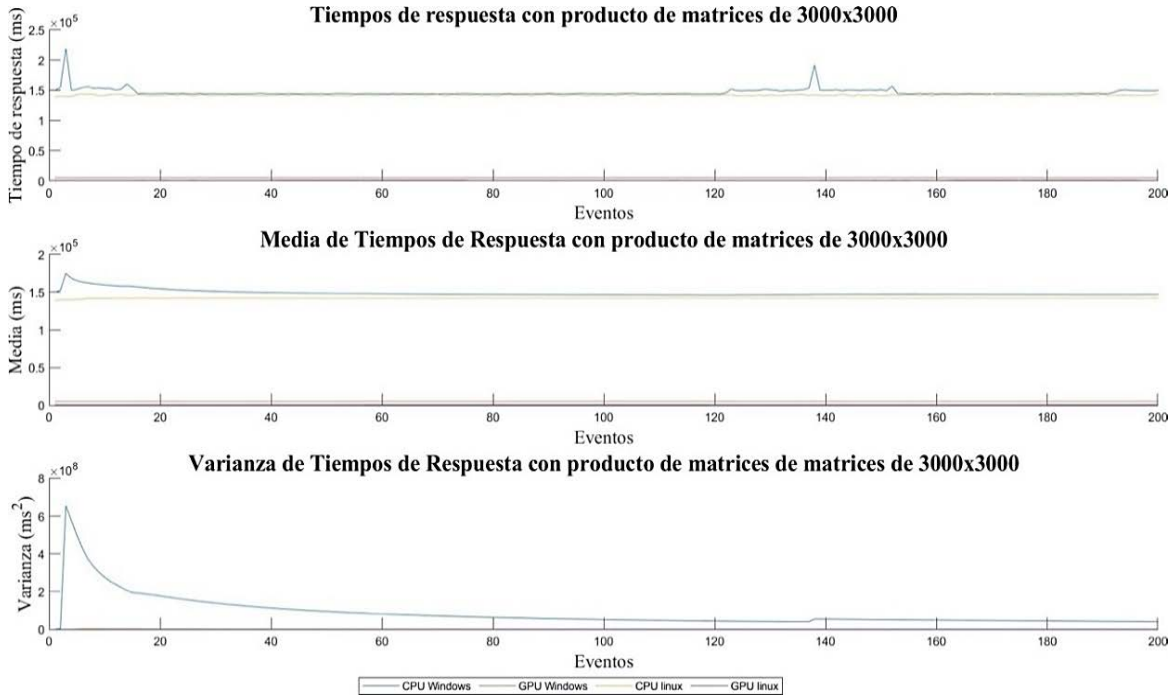

Figura 15 Tiempos de respuesta con producto de matrices de 3000×3000 (GPU integrada).

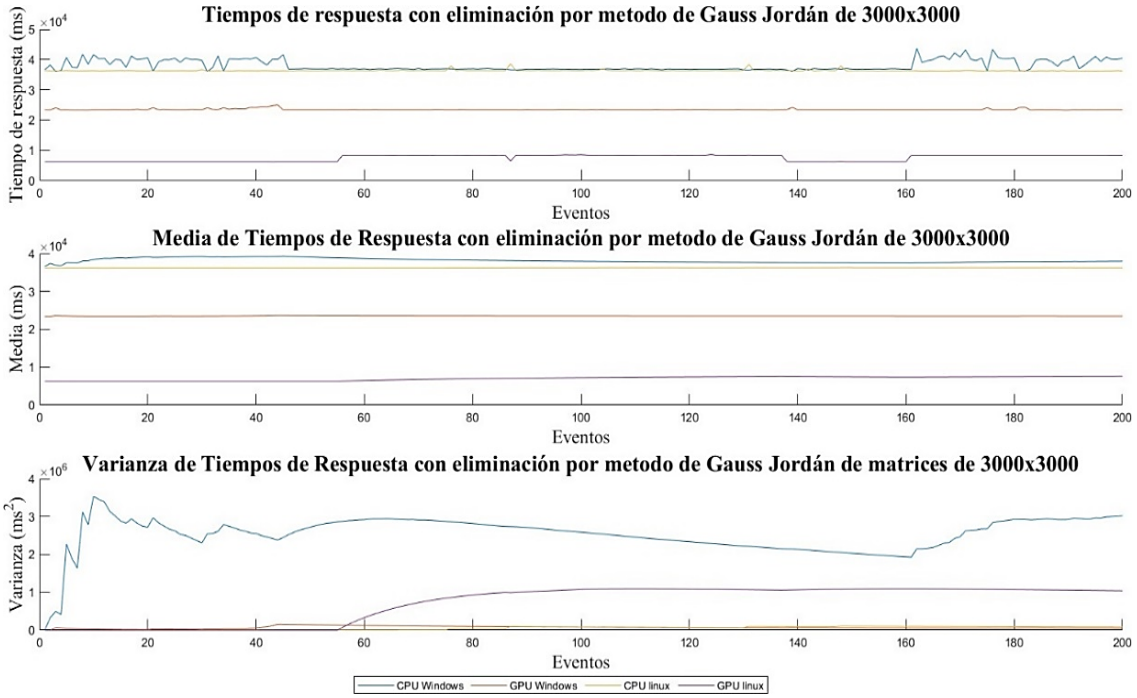

Figura 16 Tiempos de respuesta con eliminación por método de Gauss-Jordan de matrices de 3000×3000 (GPU integrada).

4. Discusión

De acuerdo con los resultados mostrados En figuras13 a 16, se seleccionó la matriz de dimensión 3000×3000, ya que durante la experimentación se observa que el uso de la memoria compartida tiene un mejor desempeño con matrices de grandes dimensiones, pues con una matriz pequeña al realizar pocas operaciones, solo consume tiempo guardando los resultados de las operaciones.

Los primeros momentos de probabilidad muestran la dinámica global de los tiempos de respuesta, estos elementos permiten analizar su desempeño, ya que a través de la media se conoce la velocidad del tiempo de cálculo en las distintas plataformas. La varianza, tiene un gran impacto para verificar qué tan confiable es ejecutar procesos complejos que requieran alto poder de cómputo.

Es importante destacar que los resultados comprueban que la paralelización de código en CPU no es una opción, ya que el CPU constantemente atiende interrupciones del sistema a esto se le atribuyen la alta demanda temporal, así como varianzas elevadas. Dicho lo anterior, analizando las gráficas resulta evidente que utilizando GPU con el sistema operativo GNU/Linux, se obtienen tiempos de respuesta menores, así como varianzas mínimas, además de no presentarse variaciones considerables. Se han seleccionado las figuras 14 y 16 pertenecientes al algoritmo eliminación de matrices por Gauss-Jordan para mostrar en la tabla 5 una comparativa con base en sus cualidades observadas.

Tabla 5 Cualidades del comportamiento de tiempo de respuesta generado mediante la ejecución del algoritmo eliminación por el método de Gauss-Jordan.

GPU externa. NVIDIA GeForce GTX 1050 externa a equipo Lenovo			
Sistema operativo	**Tiempo de respuesta aproximado**	**Comportamiento en media**	**Comportamiento en varianza**
GNU/LINUX	$0.7x10^4$ ms	Estacionaria en todos los eventos	Ninguna variación considerable, convergente a cero
WINDOWS 10	$2.8x10^4$ ms	Ligero pico en primeros eventos	Pico alto de varianza aproximado a $5.8x10^8$ ms^2 durante el primer evento
GPU integrada. Equipo HP Pavilion con GPU NVIDIA GeForce GTX 1050			
GNU/LINUX	$0.7x10^4$ ms	Ligeros aumentos intermitentes de tiempo de respuesta a partir del evento 60	Se observa ligero aumento de variación a partir del evento 60, esta variación puede deberse a interrupciones del sistema
WINDOWS 10	$2.3x10^4$ ms	Ligeros picos durante los primeros 45 eventos	Variación ligera en primeros 45 eventos, su varianza se aproxima a cero durante la evolución de los eventos

De la tabla 5 se observa que el mejor tiempo de respuesta se obtiene con GNU/Linux; sin embargo, a pesar de que éste administra de forma adecuada sus recursos, puede a llegar a ser sensible a alteraciones por llamadas o interrupciones del sistema, es por ello que se visualiza un ligero incremento en la variación a partir del evento 60, mostrado en color morado en la figura 16.

5. Conclusiones

La reducción de los tiempos de respuesta mediante la paralelización de código con tarjetas gráficas tipo NVIDIA es una alternativa más accesible para los investigadores. El toolkit de NVIDIA y la arquitectura de las tarjetas gráficas proveen herramientas para llevar a cabo reducción paralela como lo es la memoria compartida y los contadores de hilos, a su vez existen técnicas para la reducción paralela como lo son: sincronización, divergencia de urdimbre, desarrollo de bucle, desarrollo de urdimbre y reducción dinámica.

Las gráficas muestran que el tiempo de respuesta se ve reducido dependiendo de diferentes factores, el más notorio es el sistema operativo, el tiempo de respuesta se puede optimizar al utilizar alguna técnica de reducción paralela que se adecúe al algoritmo implementado, en el algoritmo de Gauss-Jordan Jordán la técnica utilizada es la de desarrollo de bucle, mientras que en el algoritmo de producto de matrices solo se ocupan los contadores de hilos para reducir algunos bucles en el desarrollo de las operaciones.

En resumen, la mejora del tiempo de respuesta que ofrece una adaptación como ésta consta de diferentes variables, como lo son la arquitectura de la tarjeta gráfica, la cantidad de bloques e hilos de ejecución utilizados, la técnica utilizada para la optimización del algoritmo, los bloques e hilos de ejecución y el sistema operativo sobre el que se trabaja.

6. Bibliografía y Referencias

[1] Baldovinos D. L. G., Rosas J. L. C., García E. E. and López P. G. Impacto de dos Mecanismos de Planificación de Tiempo Real con RT-Linux sobre una Computadora Embebida. CIRC 2018, San Jose del Cabo, Baja California Sur,

México. México, Dirección de Publicaciones IPN: Instituto Politécnico Nacional, 2018b.

[2] Baldovinos D. L. G. Análisis Experimental de los Tiempos de Respuesta en Rt-Linux Para Una SBC. Ciudad De México, Tesis de Maestría. México, Sección de Estudios de Posgrado e Investigación. ESIME Culhuacan. Instituto Politécnico Nacional, 2018a.

[3] Baldovinos D. L. G., Rosas J. L. C., C. E. M. y Guevara López Pedro. Caracterización estadística de los tiempos de respuesta de un algoritmo complejo en RT-Linux, 2019.

[4] Ccapacca E. Q., (2012). Paralelización sobre GPU del Algoritmo de Eliminación de Gauss-Jordan para la Solución de Sistemas de Ecuaciones Lineales. Arequipa - Perú: http://www.comtel.pe/comtel2012/callforpaper2012/P40C.pdf.

[5] Corsair,(2015) Cx seriesTM modular cx650m atx fuente de alimentación modular de 650 W.: Unidades-de-fuentes-de-alimentaci%C3%B3n/cxm-series-2015-config/p/CP-9020103-NA#tab-tech-specs.

[6] G. Store, (2016). Ngff pci-e versión pcie pci-e v8.4d exp gdc estación de acoplamien- to de ordenador portátil/base de tarjeta de video portátil externa: https://es.aliexpress.com/item/4000912882833.html?spm=a2g0o.productlist.0.0.457d59ffIri89C&algo_pvid=18264914-afb8-4a9c-8b8c-006319bbe511&algo_expid=18264914-afb8-4a9c-8b8c-006319bbe511-35&btsid=zab50f0815925859895762140ed19e&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603.

[7] Gigabyte, (2020). GeForce GTX 1050 oc 2g(rev1.0/rev1.1): https://www.gigabyte.com/mx/Graphics-Card/GV-N1050OC-2GD#kf.

[8] Mercado Libre, (2020). Pci-e 8 pines a 6 + 2 pines (6 pines/8 pines power): https://articulo.mercadolibre.com.mx/MLM-700127901-pci-e-8-pines-a-6-2-pines-6-pines-8-pinespower_M?quantity=1#position=2&type=item&tracking_id=bc7ccc6d-e784-4d80-b7d4-85c3e2e571fc.

[9] Soto R. T. Programación paralela sobre arquitecturas heterogéneas. Bogotá, Colombia: Universidad Nacional de Colombia, 2016.

[10] Molina J. A. L. and Alpizar D. M. R., (2012). Análisis de rendimiento de algoritmos paralelos: https://repositorio.unan.edu.ni/1914/1/ANALISIS%20DE% 20RENDIMIENTO%20DE% 20ALGORITMOS%20PARALELOS.pdf.

[11] Koo J. J. P., Quintal L. F. C. and May O. A. C., (2016). Análisis del rendimiento de procesadores multinúcleo embebidos en dispositivos digitales al ejecutar aplicaciones paralelas: http://www.itc.mx/ojs/index.php/pistas/article/view/658.

[12] NVIDIA, (2020). Documentación del paquete de herramientas de CUDA: https://docs.nvidia.com/cuda/cuda-runtime-api/groupCUDARTEVENT.html# groupCUDART__EVENT_1g40159125411db92c835edb46a0989cd6.

[13] Universidad de Alicante, (2020). Supercomputación y sus aplicaciones para el profesional de la ingeniería informática: https://eps.ua.es/es/master-ingenieria-informatica/noticias/supercomputacion-y-sus-aplicaciones-para-el-profesional-de-la-ingenieria-informatica.html.