Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
IC3101 - Arquitectura de Computadores.

Proyecto 1.
Medición de rendimiento.

Profesor: Jorge Vargas Calvo.

Estudiantes:

María Fernanda Sanabria Solano
mfss0412@estudiantec.cr
2021005572

Andrés Valverde Barrios
andres190601@estudiantec.cr
2020248159

Sara Sinaí Segura Artavia
ssinaisa@estudiantec.cr
2019015499

Tamara Nicole Rodríguez Luna.
nicolerodriguezluna@estudiantec.cr
2021077818.

Cartago, Costa Rica.
Octubre 2021.

Resumen:

En el presente proyecto, se aborda el tema de rendimiento computacional, mediante el uso de algoritmos de rendimiento y cómo estos se utilizan para la ejemplificación de que computador es mejor a la hora de escoger uno, los mismos pueden modificar o influir en la toma de decisiones de las personas. Además de su impacto en campos como el laboral, estudiantil o personal.

Para esto se realizó una investigación tomando fundamentos de estudios e información recopilada de una gran cantidad de fuentes y bases de datos, también con código externo y código propio escrito por los estudiantes. Teniendo como resultado, que los algoritmos están involucrados en el rendimiento de los cuatro computadores estudiados que fueron sometidos a las mismas pruebas, siendo compatible entre sí.

Los resultados de dicho análisis recopilan información que posteriormente es utilizada para dar una recomendación final, calificando de menor a mayor y que puede traspasar más allá de un proyecto estudiantil a un campo donde esté en nuestras manos la elección de equipo computacional de una empresa, donde nuestra calidad de egresados de Ingeniería en Computación debe relucir con los conocimientos adquiridos en la carrera, en especial en el curso de Arquitectura de Computadores.

Estas sugerencias y recomendaciones generarían una gran influencia en los posibles usuarios llegando a condicionar sus futuros sistemas operativos a escoger, mediante el rendimiento de este.

Índice

Introducción

El rendimiento de computadores está presente en el mundo tecnológico actual, el mismo nos permite llevar a cabo una tarea computacional con gran cantidad de procesos, por ende, este debe tener un buen rendimiento que satisfaga de manera exitosa las necesidades del propietario.

Por otro lado, frecuentemente se puede confundir con términos cercanos a la informática pero que tienen poco que ver a la hora influenciar los datos que obtenemos al exponer el computador a pruebas, mientras que con los resultados se demuestra la diferencia entre los computadores que, a pesar de ser compatibles a la hora de ser sometidos a la prueba, los valores que retornan los algoritmos nos darán una amplia vista sobre lo que realmente pasa internamente mientras funciona un computador con un ejecutable.

Una persona en conjunto con prestigio profesional puede ser afectada de gran manera, por una mala recomendación ante una empresa en temas de rendimiento computacional, el que la elección de un profesional no tenga peso en pruebas contundentes hace que nuestra credibilidad se vuelva desconfianza.

Marco teórico y conceptual

1. ALU:

La Unidad Aritmética Lógica (ALU) es el más importante componente de los procesadores. Todos los cálculos aritméticos y lógicos. Se realizan dentro de la ALU.

ALU es la unidad aritmética fundamental que contiene fijos multiplicadores de puntos, sumador, producto escalar, comparador, absoluto valores y operaciones lógicas. Multiplicador paralelo y sumador paralelo en la ALU son el método aritmético clave para la velocidad en procesadores. La velocidad de un procesador depende de la arquitectura del hardware, el retardo y la potencia. A alta velocidad los procesadores requieren multiplicadores de alta velocidad. (FPGA realization of ALU for mobile GPU, 2016)

2. FPU:

La FPU o Floating Point Unit es la responsable de hacer las operaciones matemáticas, ya sean sumas, restas, multiplicaciones, divisiones u otras, que requieran o utilicen números punto flotantes en una aplicación o programa. Esta unidad es usualmente utilizada a la hora de usar ecuaciones trigonométricas. *(Peter Barry & Patrick Crowley, 2012)*

3. GPU:

La GPU es un procesador formado por muchos núcleos más pequeños y especializados. Al trabajar conjuntamente, los núcleos ofrecen un desempeño masivo cuando se puede dividir una tarea de procesamiento y es procesada por muchos núcleos. *(Intel, s.f)*

Objetivo General

- Consolidar los conocimientos sobre medición de rendimiento de computadoras.

Objetivos Específicos

- Introducir a la aplicación de métodos experimentales al trabajo del ingeniero en computación.

Desarrollo

Rendimiento de la ALU

Descripción del ambiente específico en el que se lleva a cabo la comparación.

Los ambientes específicos utilizados en este rendimiento son cuatro computadores con las siguientes características:

1. Computador A:

    Dueño: Andrés Valverde Barrios.

    Sistema operativo: Windows 10 Home Edition.

    Versión: 64-bit

    Espacio disponible en disco:256GB (94 GB libres) SSD y 1TB(724 GB libres) HDD.

    Procesador: AMD Ryzen 7 2700X 3.7 GHz, 8 cores, 16 hilos.

    GPU: NVIDIA RTX 2070 8GB

    Memoria RAM: 16 gb de ram 2666 hz.

    Antigüedad: 2 años.

2. Computador B:

    Dueña: María Fernanda Sanabria Solano

    Sistema operativo: Windows 10 Pro

    Versión: 64-bit

    Espacio disponible en disco: 962GB (755 GB libres) SSD

    Procesador: Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz 2.40 GHz

    GPU: NVIDIA GeForce GTX 1650 Ti 4GB

    Memoria RAM: 32.0 GB

    Antigüedad: 10 meses

3. Computador C:

    Dueña: Sara Segura Artavia

    Sistema operativo: Windows 10 Home

    Versión: 64-bit

    Espacio disponible en disco: 893 GB (567 GB libres) SSD

Procesador: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz 4 cores, 8 hilos

GPU: NVIDIA GeForce MX130 4GB

Memoria RAM: 8GB

Antigüedad: 3 años

4. Computador D:

Dueña: Tamara Nicole Rodríguez Luna.

Sistema operativo: Windows 10 Home Single.

Versión: Sistema operativo de 64 bits, procesador basado en x64.

Espacio disponible en disco: 1,65 TB libres de 1,81 TB en disco HDD.

Procesador: AMD Ryzen 5 3500U con Radeon Vega Mobile Gfx 2.10 GHz

GPU: AMD Radeon(TM) Vega 8 Graphics, 2 GB.

Memoria RAM: 12,0 GB (9,94 GB usable).

Antigüedad: 1 año y 6 meses.

Justificación de que los ambientes de las máquinas a comparar son compatibles.

Las cuatro computadoras poseen propiedades similares con respecto al espacio de los discos duros, el sistema operativo y el tiempo que estás llevan en uso. Aún cuando estas características no son exactas en las cuatro computadoras, no poseen una diferencia tan exagerada que pueda provocar algún error a la hora de realizar las pruebas.

Se tomó la decisión de que las cuatros máquinas usarán el editor de Visual Studio Code. Además, en este se descargaron para usar la misma extensión de Python para la ejecución de las pruebas y de esta manera no afectar los resultados por la utilización de IDLES distintos en los computadores.

Los enlaces para ver nuestra versión de Visual Studio Code y la extensión de Python son:

Descarga del Visual Studio Code usado en las cuatro computadoras:

Documentation for Visual Studio Code

Descarga de la extensión de Python implementada en Visual Studio Code:

Python - Visual Studio Marketplace

Definición de lo que se está midiendo.

La Unidad Aritmética Lógica (ALU) es el más importante componente de los procesadores. Todos los cálculos aritméticos y lógicos. Se realizan dentro de la ALU.

> ALU es la unidad aritmética fundamental que contiene fijos multiplicadores de puntos, sumador, producto escalar, comparador, absoluto valores y operaciones lógicas. Multiplicador paralelo y sumador paralelo en la ALU son el método aritmético clave para la velocidad en procesadores. La velocidad de un procesador depende de la arquitectura del hardware, el retardo y la potencia. A alta velocidad los procesadores requieren multiplicadores de alta velocidad. (FPGA realization of ALU for mobile GPU, 2016)

Técnica escogida de medición

Para comparar el rendimiento del ALU, se decidió utilizar la comparación y ordenamiento de números.

Justificación de la técnica elegida

Para la comparación de números, la ALU utiliza un componente llamado digital comparator (comparador digital) para recibir como entradas los números a comparar en forma binaria y por medio de las compuertas lógicas(AND, OR, NAND, NOR, XOR, etc.) retorna como salida el resultado de la comparación.

Descripción de la herramienta seleccionada

Para le medición de la eficiencia del ALU, se utilizó una herramienta que produce varias listas de diversas longitudes con valores numéricos aleatorios como su contenido, una vez se generan estas listas, utilizamos dos tipos de algoritmos de ordenamiento, merge y quick, para organizar los valores de las listas de menor a mayor. El mismo programa produce como parte de sus salidas el tiempo que tomó cada algoritmo en ordenar la lista de X longitud, lo cual facilitó la comparación de rendimientos.

El algoritmo Merge Sort utiliza las mitades de las listas y las ordena antes de reintegrarse a la lista principal, es un algoritmo de tipo divide and conquer, por lo que nos permite a su vez medir el rendimiento del equipo en procesos que requieran recursividad.

El algoritmo de clasificación por fusión divide la matriz en dos mitades y las clasifica por separado. Después de ordenar las dos mitades de la matriz, las fusiona en una sola matriz ordenada. Como es un algoritmo recursivo, divide la matriz hasta que la matriz se convierte en la más simple (matriz con un elemento) para ordenar.

Veamos los pasos para implementar el tipo de fusión:

1. Inicializa la matriz con datos ficticios (enteros).
2. Escribe una función llamada unir para fusionar submatrices en una sola matriz ordenada.
3. Acepta la matriz de argumentos, los índices izquierdo, medio y derecho.
4. Obtiene las longitudes de las submatrices izquierda y derecha utilizando los índices dados.
5. Copia los elementos de la matriz en las respectivas matrices izquierda y derecha.
6. Repite las dos submatrices.
7. Compara los dos elementos de las submatrices.
8. Reemplaza el elemento de la matriz con el elemento más pequeño de las dos submatrices para ordenar.
9. Comprueba si quedan elementos en ambas submatrices.
10. Los agrega a la matriz.
11. Escribe una función llamada merge_sort con matriz de parámetros, índices izquierdo y derecho.
12. Si el índice de la izquierda es mayor o igual que el índice de la derecha, regresa.
13. Encuentra el punto medio de la matriz para dividir la matriz en dos mitades.
14. Llama recursivamente al merge_sort utilizando los índices izquierdos, derecho y medio.
15. Después de las llamadas recursivas, combina la matriz con el unir función.

(Kareem, 2021)

El algoritmo Quicksort es similar al merge, ya que primero ordena trozos de la lista, estas particiones se realizan en torno a un pivote, la cual divide la lista en dos secciones

diferentes y las ordena, resultando la lista izquierda en elementos menores al pivote y a la derecha los elementos que son mayores a este.

En la práctica, es el algoritmo de ordenación más rápido conocido, su tiempo de ejecución promedio es O (n log (n)), siendo en el peor de los casos O(n2), caso altamente improbable. El hecho de que sea más rápido que otros algoritmos de ordenación con tiempo promedio de O (n log (n)) (como SmoothSort o HeapSort) viene dado por que QuickSort realiza menos operaciones ya que el método utilizado es el de partición.

Explicación abstracta del funcionamiento de QuickSort:

1. Se elige un elemento v de la lista L de elementos al que se le llama pivote.
2. Se particiona la lista L en tres listas:

L1 - que contiene todos los elementos de L menos v que sean menores o iguales que v.

L2 - que contiene a v.

L3 - que contiene todos los elementos de L menos v que sean mayores o iguales que v

3. Se aplica la recursión sobre L1 y L3.
4. Se unen todas las soluciones que darán forma final a la lista L finalmente ordenada. Como L1 y L3 están ya ordenados, lo único que tenemos que hacer es concatenar L1, L2 y L3
5. Aunque este algoritmo parece sencillo, hay que implementar los pasos 1 y 3 de forma que se favorezca la velocidad de ejecución del algoritmo.

La velocidad de ejecución del algoritmo depende en gran medida de cómo se implementa este mecanismo, una mala implementación puede suponer que el algoritmo se ejecute a una velocidad mediocre o incluso pésima. La elección del pivote determina las particiones de la lista de datos, por lo tanto, huelga decir que esta es la parte más crítica de la implementación del algoritmo QuickSort. Es importante intentar que al seleccionar el pivote v las particiones L1 y L3 tengan un tamaño idéntico dentro de lo posible (Campos, 2011.)

Descripción de la herramienta seleccionada

El código fuente creado tiene las funciones extra de medir tiempo, generar lista y prueba para poder tener el resultado del rendimiento con los valores N.

https://gist.github.com/nottwithtt/cf64c08aa525e7b3ac406f9717dbd4e0

Justificación de que la herramienta seleccionada implementa correctamente la técnica escogida.

El código utilizado como herramienta es interno, el cual fue creado por una de los integrantes del proyecto Tamara Nicole Rodríguez Luna, el algoritmo utilizado nos permite implementar de una manera flexible y simple la técnica de nuestra escogencia, ya que no solo nos permite verificar que los resultados han sido ordenados de la manera correcta, sino que también implementa diversos tipos de algoritmos de ordenamiento y utiliza los mismos parámetros a través de estos para medir la eficiencia del ALU en diversas implementaciones.

Descripción de los experimentos realizados

Se instaló en las 4 máquinas el IDE Visual Studio Code versión 1.60 y se utilizó la misma extensión de Python para poder ejecutar el código, esto para así eliminar factores externos que pudieran modificar el tiempo de ejecución, después, se procedió a realizar 14 corridas del código con longitudes de lista de 100, 500, 1000, 5000, 10000, 50000. Una vez que obtuvimos el tiempo de ejecución de estos procesos se procedió a trasladarlos a una hoja Excel donde se pudieron representar por medio de gráficas.

Justificación de los parámetros usados para los experimentos

Se realizaron pruebas de ordenamientos de números enteros random con listas de tamaño 100, 500, 1000, 5000, 10000, 50000 para probar con distintas cantidades. No se usaron listas menores para poder comparar con tiempos más grandes y de una variabilidad más aceptable; tampoco se usaron listas mayores debido a que durarán más tiempo del necesario atrasándonos a la hora de hacer las repeticiones y la variabilidad de valores afectaría los promedios restándoles exactitud.

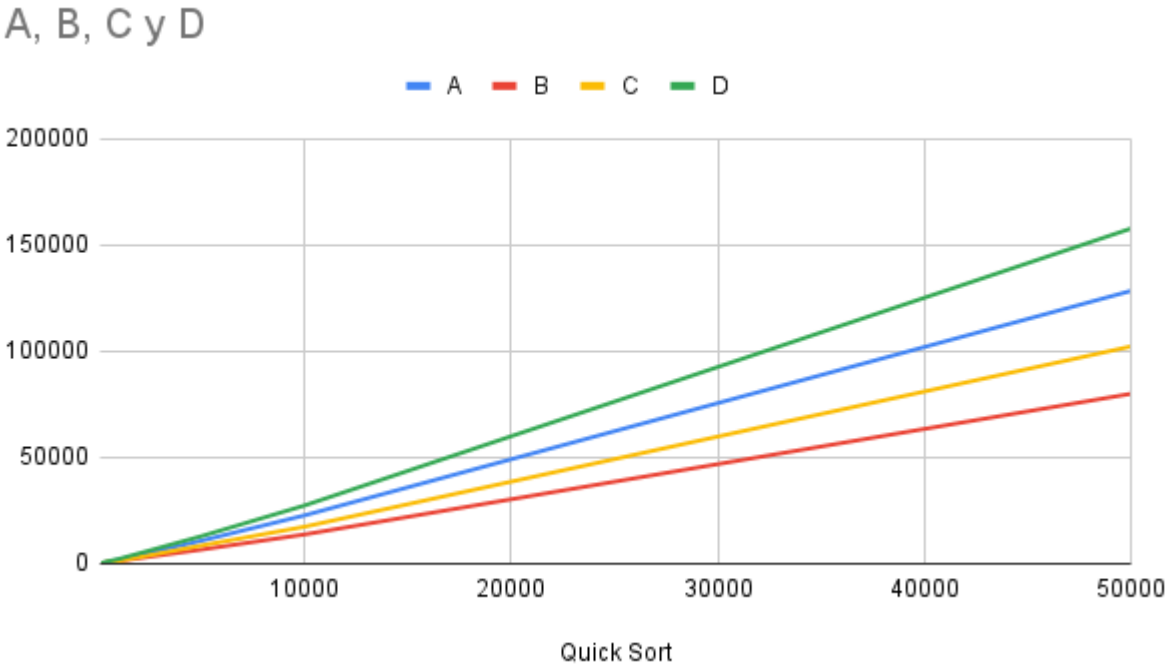
Con respecto a las repeticiones, para calcular el número de estas se tomó como límite máximo de pruebas 100 pruebas, ya que más de estas podría distorsionar los resultados debido al largo uso continúo de la computadora al realizar las pruebas. Después se discutió el margen de error aceptando finalmente un 25%, perdiendo razonablemente la exactitud, pero no sobrecargando los recursos de la computadora.

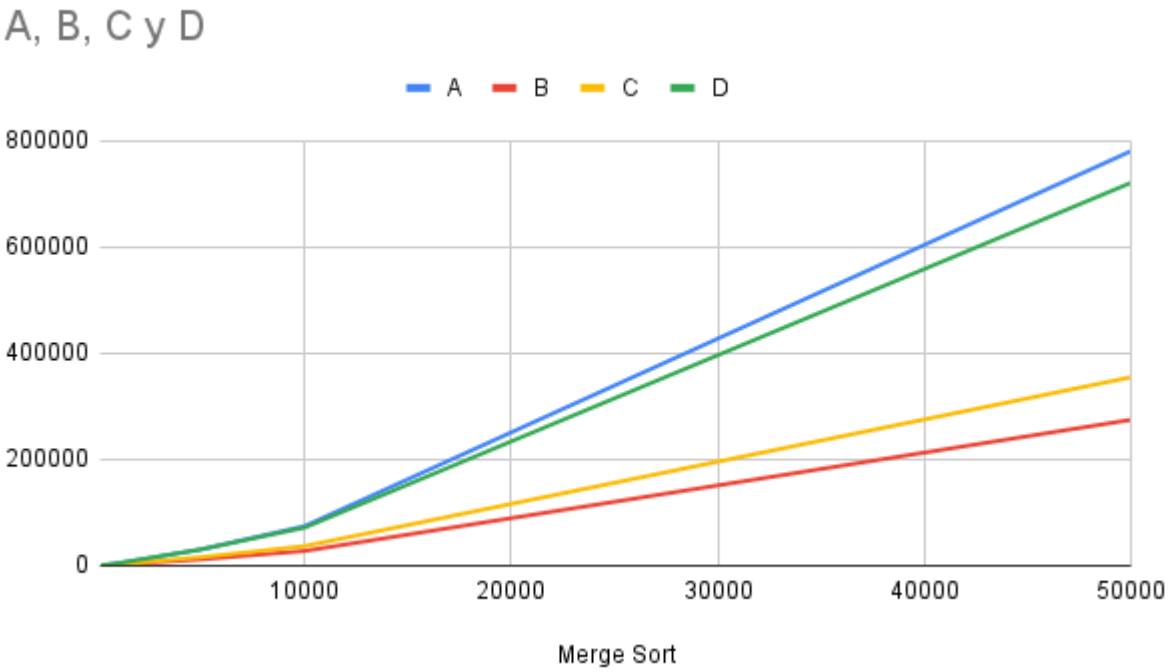
Por último, se acordó un nivel de confianza de 95%, estando entre las posibilidades un 95% o un 99%. Tomando en cuenta estas variables se calculó el tamaño de la muestra de manera estadística resultando una muestra de 14 pruebas por tamaño de lista.

Resultados empíricos obtenidos

Ver Apéndice A.

Gráficos que resumen los resultados empíricos





Justificación de la técnica gráfica empleada

Se aplicaron las gráficas de líneas para representar los promedios de los resultados de las catorce corridas de código en las cuatro computadoras con listas de valores 100, 500, 1000, 5000, 10000 y 50000 en ambos métodos (Quicksort y Merge Sort). Esta gráfica fue utilizada para mostrar de manera activa y directa al usuario la comparación entre los tiempos en milisegundos que duraron las computadoras en hacer los ordenamientos.

Rendimiento de la FPU

Descripción del ambiente específico en el que se lleva a cabo la comparación

Los ambientes específicos utilizados en este rendimiento son cuatro computadores con las siguientes características:

1. Computador A:

   Dueño: Andrés Valverde Barrios.

   Sistema operativo: Windows 10 Home Edition.

   Versión: 64-bit

   Espacio disponible en disco:256GB (94 GB libres) SSD y 1TB(724 GB libres) HDD.

   Procesador: AMD Ryzen 7 2700X 3.7 GHz, 8 cores, 16 hilos.

   GPU: NVIDIA RTX 2070 8GB

   Memoria RAM: 16 gb de ram 2666 hz.

   Antigüedad: 2 años.

2. Computador B:

   Dueña: María Fernanda Sanabria Solano

   Sistema operativo: Windows 10 Pro

   Versión: 64-bit

   Espacio disponible en disco: 962GB (755 GB libres) SSD

   Procesador: Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz 2.40 GHz

   GPU: NVIDIA GeForce GTX 1650 Ti 4GB

   Memoria RAM: 32.0 GB

   Antigüedad: 10 meses

3. Computador C:

   Dueña: Sara Segura Artavia

   Sistema operativo: Windows 10 Home

   Versión: 64-bit

   Espacio disponible en disco: 893 GB (567 GB libres) SSD

   Procesador: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz 4 cores, 8 hilos

GPU: NVIDIA GeForce MX130 4GB

Memoria RAM: 8GB

Antigüedad: 3 años

4. Computador D:

Dueña: Tamara Nicole Rodríguez Luna.

Sistema operativo: Windows 10 Home Single.

Versión: Sistema operativo de 64 bits, procesador basado en x64.

Espacio disponible en disco: 1,65 TB libres de 1,81 TB en disco HDD.

Procesador: AMD Ryzen 5 3500U con Radeon Vega Mobile Gfx 2.10 GHz

GPU: AMD Radeon(TM) Vega 8 Graphics, 2 GB.

Memoria RAM: 12,0 GB (9,94 GB usable).

Antigüedad: 1 año y 6 meses.

Justificación de que los ambientes de las máquinas a comparar son compatibles

Las cuatro computadoras poseen propiedades similares con respecto al espacio de los discos duros, el sistema operativo y el tiempo que estás llevan en uso. Aún cuando estas características no son exactas en las cuatro computadoras, no poseen una diferencia tan exagerada que pueda provocar algún error a la hora de realizar las pruebas.

Se tomó la decisión de que las cuatros máquinas usarán el editor de Visual Studio. Además, en este se descargaron y usó la misma extensión para la ejecución de las pruebas y de esta manera no afectar los resultados por la utilización de editores o compiladores diferentes.

Descarga del Visual Studio Code usado en las cuatro computadoras:

Documentation for Visual Studio Code

Descarga de la extensión de Java implementada en Visual Studio Code:

https://code.visualstudio.com/docs/languages/java

Definición de lo que se está midiendo

La FPU o Floating Point Unit es la responsable de hacer las operaciones matemáticas, ya sean sumas, restas, multiplicaciones, divisiones u otras, que requieran o utilicen números punto flotantes en una aplicación o programa. Esta unidad es usualmente utilizada a la hora de usar ecuaciones trigonométricas. *(Peter Barry & Patrick Crowley, 2012)*

Técnica escogida de medición

Utilizar un código para lograr medir el tiempo que se tarda al hacer el cálculo de operaciones que utilicen la unidad punto flotante haciendo uso de valores punto flotante, tanto positivos como negativos, y aplicando ecuaciones trigonométricas al cálculo.

Calcular una definida cantidad de repeticiones las medidas necesarias para diseñar un telescopio acromático de 4 pulgadas utilizado como ejemplo de la obra clásica de Wyld sobre trazados a mano presentado en Amateur Telescope Making, Volumen 3.

Justificación de la técnica elegida

Los cálculos que requieran valores punto flotantes, tanto positivos como negativos, requieren de la unidad punto flotante para obtener un resultado acertado. Además, el uso de ecuaciones trigonométricas utiliza igualmente la unidad punto flotante para estas llevarse a cabo.

Midiendo el tiempo que le toma a las computadoras producir los resultados de cálculos que utilizan valores punto flotantes y que implementan operaciones trigonométricas nos permite comparar el rendimiento de las unidades punto flotantes de las computadoras.

Descripción de la herramienta seleccionada

El código utilizado permite medir el tiempo que se tarda al calcular una definida cantidad de repeticiones las medidas necesarias para diseñar un telescopio acromático de 4 pulgadas utilizado como ejemplo de la obra clásica de Wyld sobre trazados a mano presentado en Amateur Telescope Making, Volumen 3.

Código fuente:

https://gist.github.com/nottwithtt/219f299abd67bf4d04f2ad971bd65220

Justificación de que la herramienta seleccionada implementa correctamente la técnica escogida

Las ecuaciones utilizadas en el código para el cálculo de las medidas requeridas para el diseño de un telescopio acromático de 4 pulgadas requieren de valores punto flotantes, usando pi (π) como un ejemplo de estos, además se aplican ecuaciones trigonométricas que utilizan la unidad punto flotante.

Descripción de los experimentos realizados

Se realizaron 14 corridas del código anteriormente descrito para obtener un average entre los resultados de lo que la computadora tarda en milisegundos para realizar 100, 500, 1000, 5000, 10000 y 50000 iteraciones del cálculo de las medidas requeridas para el diseño de un telescopio acromático de 4 pulgadas. Estos resultados fueron anotados y posteriormente comparados entre las cuatro computadoras.

Justificación de los parámetros usados para los experimentos

Al igual que en el experimento de la ALU, se realizaron pruebas de longitudes 100, 500, 1000, 5000, 10000 y 50000, no se utilizaron listas menores a 100 debido a que la diferencia entre estas era muy pequeña, tanto al punto que los resultados no eran relevantes a los gráficos y los resultados.

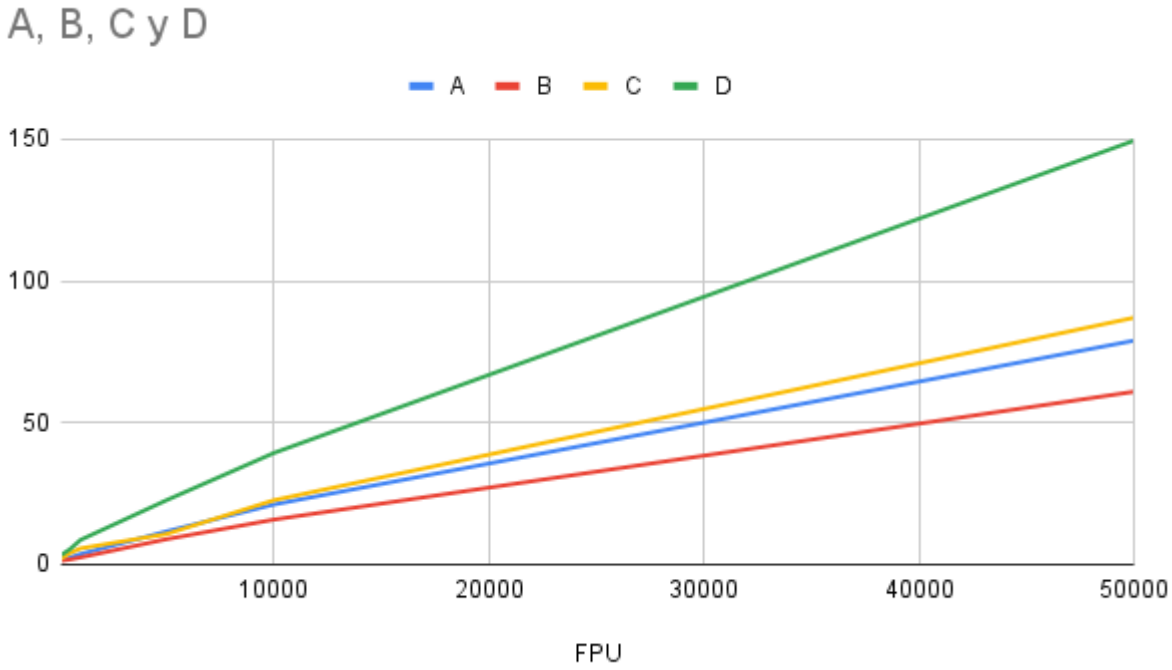
Los valores que utilizamos aumentan exponencialmente, ya que consideramos que para que los resultados entre las longitudes fueran significativos tenía que existir una diferencia notable entre estos. El valor más grande utilizado fue de 50000, este es un valor que significa una carga considerable a la FPU sin necesidad de extender demasiado el tiempo de ejecución o restarle relevancia a los otros resultados.

Por otra parte, se decidió que la cantidad de corridas fueran 14 ya que queríamos que se mantuvieran la menor cantidad de diferencias entre las pruebas de la ALU y el FPU, por lo que se mantuvo el margen de error del 25% y un nivel de confianza del 95%.

Resultados empíricos obtenidos

Ver apéndice A.

Gráficos que resumen los resultados empíricos.



Justificación de la técnica gráfica empleada

Se utilizó la gráfica anterior para mostrar de manera dinámica los promedios de los resultados de las computadoras A, B, C y D en las pruebas de FPU para los distintos valores de iteraciones: 100, 500, 1000, 5000, 10000 y 50000.

Se emplearon las gráficas de líneas debido a que estas son usadas regularmente para mostrar tendencias en distintos grupos de datos; de esta manera podemos probar cuánto aumenta el tiempo en las computadoras al ir aumentando el número de iteraciones a realizar y poder hacer comparaciones entre esto.

Rendimiento de la GPU

Descripción del ambiente específico en el que se lleva a cabo la comparación

Los ambientes específicos utilizados en este rendimiento son cuatro computadores con las siguientes características:

1. Computador A:

   Dueño: Andrés Valverde Barrios.

   Sistema operativo: Windows 10 Home Edition.

   Versión: 64-bit

   Espacio disponible en disco:256GB (94 GB libres) SSD y 1TB(724 GB libres) HDD.

   Procesador: AMD Ryzen 7 2700X 3.7 GHz, 8 cores, 16 hilos.

   GPU: NVIDIA RTX 2070 8GB

   Memoria RAM: 16 gb de ram 2666 hz.

   Antigüedad: 2 años.

2. Computador B:

   Dueña: María Fernanda Sanabria Solano

   Sistema operativo: Windows 10 Pro

   Versión: 64-bit

   Espacio disponible en disco: 962GB (755 GB libres) SSD

   Procesador: Intel(R) Core(TM) i9-10885H CPU @ 2.40GHz 2.40 GHz

   GPU: NVIDIA GeForce GTX 1650 Ti 4GB

   Memoria RAM: 32.0 GB

   Antigüedad: 10 meses

3. Computador C:

   Dueña: Sara Segura Artavia

   Sistema operativo: Windows 10 Home

   Versión: 64-bit

   Espacio disponible en disco: 893 GB (567 GB libres) SSD

   Procesador: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz 4 cores, 8 hilos

GPU: NVIDIA GeForce MX130 4GB

Memoria RAM: 8GB

Antigüedad: 3 años

4. Computador D:

Dueña: Tamara Nicole Rodríguez Luna.

Sistema operativo: Windows 10 Home Single.

Versión: Sistema operativo de 64 bits, procesador basado en x64.

Espacio disponible en disco: 1,65 TB libres de 1,81 TB en disco HDD.

Procesador: AMD Ryzen 5 3500U con Radeon Vega Mobile Gfx 2.10 GHz

GPU: AMD Radeon(TM) Vega 8 Graphics, 2 GB.

Memoria RAM: 12,0 GB (9,94 GB usable).

Antigüedad: 1 año y 6 meses.

Justificación de que los ambientes de las máquinas a comparar son compatibles.

Las cuatro computadoras poseen propiedades similares con respecto al espacio de los discos duros, el sistema operativo y el tiempo que estás llevan en uso. Aun cuando estas características no son exactas en las cuatro computadoras, no poseen una diferencia tan exagerada que pueda provocar algún error a la hora de realizar las pruebas.

Dos de las máquinas comparten el procesador de Intel con cores distintas y las otras dos comparten el procesador Ryzen con distintas versiones. Lo anterior permitió una comparación fluida entre procesadores distintos y procesador de distinta versión sin que otros elementos como la capacidad RAM o del disco duro intervinieran o alterarán los resultados. Al ser este trabajo con cuatro computadores, los mismos descargaron la misma versión de código abierto lo que permite la transparencia de los resultados para este proyecto.

Definición de lo que se está midiendo.

La GPU es un procesador formado por muchos núcleos más pequeños y especializados. Al trabajar conjuntamente, los núcleos ofrecen un desempeño masivo cuando se puede dividir una tarea de procesamiento y es procesada por muchos núcleos. (Intel, s.f)

Piccoli (2011) afirma: “En el caso de las arquitecturas many-cores, los desarrollos se centran en optimizar el desempeño de las aplicaciones. Dentro de este tipo de arquitectura

se encuentran las están las Unidades de Procesamiento Gráfico (Graphics Processing Unit, GPU).” (pág. 31)

Además, agrega:

> Las unidades de procesamiento gráfico, GPU, se han convertido en una parte integral de los sistemas actuales de computación. El bajo costo y marcado incremento del rendimiento permitieron su fácil incorporación al mercado. En los últimos años, su evolución implicó un cambio, dejó de ser un procesador gráfico potente para convertirse en un co-procesador apto para el desarrollo de aplicaciones paralelas de propósito general con demanda de anchos de banda de procesamiento y de memoria sustancialmente superiores a los ofrecidos por la CPU.
>
> La rápida adopción de las GPU como computadora paralela de propósito general se vio favorecida por el incremento tanto de sus capacidades como de las facilidades y herramientas de programación. Actualmente la GPU se ha posicionado como una alternativa atractiva a los sistemas tradicionales de computación paralela. (pág. 30)

Por otro lado, Juega afirma:

> En la actualidad las plataformas multicore lideran la industria de los computadores, obligando a los desarrolladores de software a adaptarse a nuevos paradigmas de programación para poder explotar su capacidad de cómputo. A día de hoy uno de los principales exponentes de las plataformas multiforme son las unidades de procesamiento gráfico (GPUs).
>
> El desarrollo de aplicaciones para GPU requiere un alto esfuerzo por parte de los programadores. Por un lado, deben modelar los problemas de modo que permitan el aprovechamiento de plataformas masivamente paralelas. Por otro lado, deben preocuparse de que las aplicaciones hagan un uso eficiente del sistema de memoria, heterogéneo, de múltiples niveles, con gestión software o hardware. En general, dado un algoritmo concreto, el espacio de soluciones posibles para un mapeo sobre GPU es enorme. El recurso habitual de los programadores es el ensayo, prueba y error de distintas soluciones, guiados solo por su propia experiencia e intuición, lo que resulta ineficiente de cara al desarrollo y mantenimiento de software. (pág. 3)

Técnica escogida de medición.

Según un estudio realizado para la Asociación Argentina de Mecánica Computacional por Christian Pérez y Fabiana Piccoli, se dice que hay muchas formas de calificar una GPU, ya sea a través de pruebas cuantitativas, como el tiempo de respuesta, o cualitativas, como la confiabilidad. Además, Pérez menciona que para medir el rendimiento la mayor parte del tiempo se usan las pruebas de estrés, la cual fue la escogida para este proyecto, complementada por una cuantitativa a través de benchmarking. (2010)

Las pruebas de estrés son evaluaciones de rendimiento de alta intensidad, generalmente instrumentadas, es decir, hechas a través de programas adicionales hechos especialmente para llevar un componente, en este caso la GPU, a su máxima capacidad y probar si el estado en el que se encuentra es adecuado. (IBM, s.f)

Los benchmarks son definidos por la empresa Intel (2014) como programas computacionales especializados que se ejecutan en los sistemas que serán evaluados, para luego ser comparados, ya que estos ejecutan varias pruebas y generan un puntaje de desempeño, el cual permite una comparación objetiva.

Justificación de la técnica elegida

Se eligió realizar pruebas de estrés ya que estas permiten medir la velocidad de respuesta, la capacidad y el desempeño de un componente poniéndolo al máximo con diferentes pruebas. Generalmente los componentes como la GPU no estarán trabajando al 100% de su capacidad, sin embargo, pruebas como esta la llevan a su límite para averiguar su máxima capacidad y su respuesta a este, por ello permite saber su estado y rendimiento al compararlo con otros datos, ya sea de internet o con el de otros equipos.

Descripción de la herramienta seleccionada

La herramienta escogida para realizar las pruebas de estrés fue Phoronix Test Suite en su versión 10.4.0. Este programa fue creado en el 2004 originalmente para ser usado en Linux, sin embargo, con los años se continuó su desarrollo para dar soporte a Windows y Mac. Al ser un software de código abierto permite también que otras personas trabajen sobre una copia del código original y hagan cambios, añadan funcionalidades, entre otros.

Este programa permite hacer más de 100 pruebas desarrolladas por diferentes empresas, e incluso desarrolladores independientes, tanto del GPU como del procesador, y evaluar su rendimiento. Su código fuente se encuentra en el siguiente repositorio:

https://github.com/phoronix-test-suite/phoronix-test-suite

Justificación de que la herramienta seleccionada implementa correctamente la técnica escogida

La herramienta escogida es externa y de código abierto, mediante la cual podemos confiar en los resultados de las pruebas de estrés.

Descripción de los experimentos realizados.

Test utilizado: FurMark.

El test utiliza medición de los gráficos del computador, donde tiene un Average Install Time de 3 segundos y un Average Run Time de 3 minutos con 18 segundos.

Según los datos de OpenBenchmarking.org, la prueba seleccionada/ la prueba de configuración de prueba seleccionada (GpuTest 0.7.0 - Prueba: Plot3D - Resolución: 1920 x 1080 - Modo: Pantalla completa) tiene un tiempo de ejecución promedio de 2 minutos. De forma predeterminada, este perfil de prueba está configurado para ejecutarse al menos 3 veces, pero puede aumentar si la desviación estándar excede los valores predeterminados predefinidos o si otros cálculos consideran necesarias ejecuciones adicionales para una mayor precisión estadística del resultado. *(Phoronix, s.f)* En la realización de este trabajo se ejecutó el mismo test una única vez dando como resultados de comparación las tres veces que por default el programa hace.

Justificación de los parámetros usados para los experimentos

La herramienta de Phoronix permite que uno elija cuántas veces se ejecuta la prueba a realizar, sin embargo, también tienen una cantidad especificada en caso de que el usuario no desee elegir, por lo que en este proyecto se usó la recomendada.

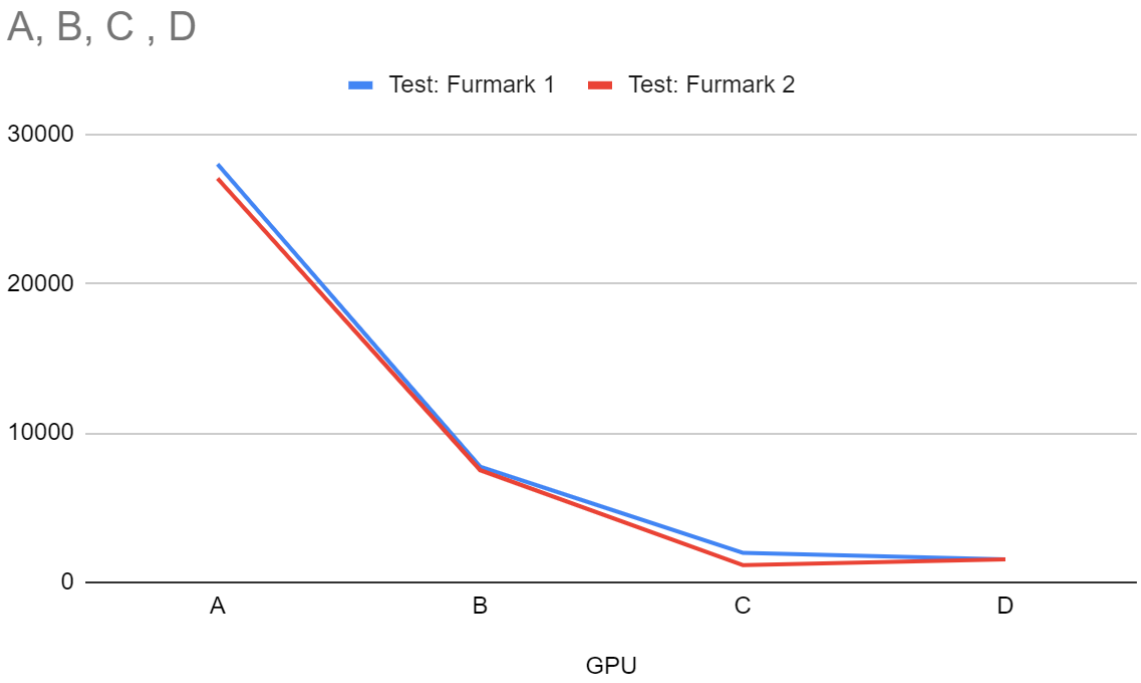
Para medir el rendimiento de la GPU se realizaron 3 corridas de la prueba en modo ventana y 3 corridas de la misma prueba, pero en modo pantalla completa. De estos resultados el programa asignó una puntuación según la velocidad de respuesta y la calidad de la imagen por cada vez que se ejecutó el programa.

Además, el programa arrojó el promedio de la puntuación de las corridas, lo cual fue el valor tomado para realizar los gráficos y comparación para la evaluación final de las computadoras, ya que demuestra el comportamiento usual aproximado al estar al máximo de su funcionamiento.

Resultados empíricos obtenidos

Ver apéndice C.

Gráficos que resumen los resultados empíricos.



Justificación de la técnica gráfica empleada.

Fue utilizada la gráfica de líneas para representar los resultados obtenidos de las pruebas realizadas del test de Furmark con la herramienta Phoronix. Esta gráfica nos muestra la diferencia entre las cuatro computadoras en los puntos obtenidos según el rendimiento de las computadoras.

Conclusiones y Recomendaciones

Durante las pruebas pudimos observar que con respecto a la ALU las que presentaron mejor rendimiento fueron las computadoras B y C. Al tomar en consideración las pruebas de la FPU y de GPU las que tienen mejor rendimiento en estas últimas son las computadoras A y B.

La computadora A tiene un precio más de 1000$, la computadora B más de los 3000$, la computadora C alrededor de los 400$ y por último la computadora D tiene un precio alrededor de los 650$. La A no es portátil.

Tomando en cuenta estos valores y los resultados obtenidos con las pruebas de rendimiento de ALU, FPU y GPU asignamos las siguientes puntuaciones a las computadoras:

El computador A recibe una calificación final de 9.

El computador B recibe una calificación final de 8.5.

El computador C recibe una calificación final de 8.

El computador D recibe una calificación final de 4.

Referencias

Barry, P., & Crowley, P. (2012). *Modern Embedded Computing.* https://www.sciencedirect.com/science/book/9780123914903

Campos, O. (2011). *Implementando el algoritmo QuickSort.*

IBM. (2021). *Pruebas de rendimiento.* https://www.ibm.com/docs/es/rtw/9.0.0?topic=phases-performance-testing

Intel. *CPU vs GPU: ¿Cuál es la diferencia? https://www.intel.la/content/www/xl/es/products/docs/processors/cpu-vs-gpu.html*

Intel. (2014). *El rol de los benchmarks en la compra de computadoras en el sector público https://www.intel.la/content/dam/www/public/lar/xl/es/documents/articles/benchmarks-spa.pdf*

Juega, C. (2010). *Estudio de rendimiento en GPU* . https://eprints.ucm.es/id/eprint/11384/1/proyecto_master.pdf

Kareem, H. (2021). *Ordenar implementaciones de algoritmos en Python. https://geekflare.com/es/python-sorting-algorithms/*

OpenBenchmarking. (2021). *GpuTest. https://openbenchmarking.org/test/pts/gputest*

Perez, C., & Piccoli, F. (2010). Estimación de los parámetros de rendimiento de una GPU.

Perez, C. & Piccoli, F. *Mecánica Computacional, XXIX* http://venus.ceride.gov.ar/ojs/index.php/mc/article/view/3224

Phoronix Media. (2021). *Open-Source, Automated Benchmarking.* https://www.phoronix-test-suite.com/?k=home

Piccoli, M. (2011). *Computación de alto desempeño en GPU*. Editorial de la Universidad Nacional de La Plata (Edulp). http://sedici.unlp.edu.ar/bitstream/handle/10915/18404/Documento_completo__.pdf?sequence=1&isAllowed=y

Tolba, M. F., Madian, A. H., & Radwan, A. G.FPGA realization of ALU for mobile GPU . Paper presented at the *International Conference on Advances in Computational Tools for Engineering Applications, ACTEA https://ieeexplore-ieee-org.ezproxy.itcr.ac.cr/document/7560104*

Apéndices

Apéndice A.

Valores N QuickSort (Andrés)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000175094604	0.000150072574	0.000175034999	0.000150024890	0.000174975395	0.000175082683	0.000150024890	1.00E-04	0.000125038623	0.000125002861	0.000150036811	0.000150060653	0.000150036811	150.0368118
500	0.000875186920	0.000875210762	0.000925207138	0.000825166702	0.000875091552	0.000900006294	0.000900173187	0.000925183296	0.000900137424	0.000875115394	0.000950181484	0.000875163078	0.000891818602	891.8186029
1000	0.001900243759	0.001850140095	0.001925265789	0.001900362968	0.001950371265	0.000900006294	0.001850390434	0.001875543594	0.001850521564	0.001925301552	0.001850354676	0.001950311681	0.001810734471	1810.734471
5000	0.0108269453	0.0105518221 9	0.0106518938	0.0106519460 7	0.0109269261 4	0.0111019253 7	0.0108268618 6	0.0108269691 5	0.0106769442 6	0.0108269929 9	0.0108269333 8	0.0110268712	0.0108102530 2	10810.25302
10000	0.0228539705 3	0.0232290983 2	0.0223790168 8	0.0227789759 6	0.0226789474 5	0.0235537171 4	0.0227540969 8	0.0223789453 5	0.0226039767 3	0.0224790215 5	0.0225289464	0.0228291034 7	0.0227539847 3	22753.98473
50000	0.1308737397	0.1281226397	0.1275724769	0.1271974921	0.128572619	0.1303980112	0.1297228813	0.1259722829	0.1276474237	0.1273223639	0.128447628	0.1289476871	0.1283997705	128399.7705

Valores N MergeSort (Andrés)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000375008583	0.000375020504	0.000350046157	0.000250041484	0.000325047969	0.000325071811	0.000325047969	0.000350105762	0.000300061702	0.000325059890	0.000350058078	0.000350034236	0.000333383679	333.3836794
500	0.002050316334	0.002075326443	0.002225387096	0.002075254917	0.002025341988	0.002025353909	0.002100312715	0.002175366879	0.002050352097	0.002075266838	0.002050352097	0.002100384235	0.002085751295	2085.751295
1000	0.004575808960	0.004550731182	0.004675817490	0.004700946808	0.004550814629	0.004552280903	0.004675745964	0.004675734043	0.004525828362	0.004601001740	0.004550874233	0.004600763321	0.004603028297	4603.028297
5000	0.030880749230	0.030605447290	0.031130492690	0.031031215190	0.030506193640	0.030631256120	0.030830538270	0.031055378910	0.030780494210	0.030830287930	0.030655288730	0.030577543642	0.030807731550	30807.73155
10000	0.074913156030	0.074763190750	0.074838352220	0.074488186840	0.073237943650	0.074288189410	0.074012959000	0.074212849140	0.074388146400	0.075013160710	0.074088060860	0.075339639190	0.074465319510	74465.31951
50000	0.78223809	0.78663872 48	0.78896440 27	0.78053747 42	0.76918571	0.78418840 17	0.77996271 85	0.77928754 09	0.77936257 12	0.77643713 95	0.78021283 15	0.77858738 9	0.78046691 62	780466.9162

Valores N QuickSort (Fer)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000149595737	4.99E-05	4.99E-05	4.98E-05	4.99E-05	0.000146842002	4.96E-05	0.000148224830	9.97E-05	9.98E-05	0.000149416923	0.000149583816	0.000099352995	99.35299555
500	0.000548577308	0.000749933719	0.000550484657	0.000648403167	0.000648331642	0.000996899604	0.000398814678	0.000548386573	0.000595498085	0.000847697258	0.000796103477	0.000498485565	0.000652301311	652.3013115
1000	0.001196944714	0.001148283482	0.001098775864	0.001149833202	0.001097023487	0.001795208454	0.001097106934	0.001396143436	0.001198172569	0.001495885849	0.001197588444	0.001029634476	0.001241716743	1241.716743
5000	0.007282912731	0.006445860863	0.006481540203	0.006589889526	0.006407010555	0.006333017349	0.006930077076	0.006511008739	0.006564474106	0.006478083134	0.007180702686	0.006335306168	0.006628323595	6628.323595
10000	0.01398772001	0.01374183893	0.01361354589	0.01378901005	0.01377820969	0.01360110044	0.01352828741	0.01364190578	0.01389687061	0.01366349459	0.01425349712	0.01376326084	0.01377156178	13771.56178
50000	0.08213375807	0.07976833582	0.07994487286	0.07906359434	0.07873278856	0.07891093493	0.0801992178	0.08073139191	0.08226335049	0.08019603491	0.07909454107	0.07854810953	0.07996557752	79965.57752

Valores N MergeSort (Fer)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000149595737	0.000199472904	0.000148034095	9.97E-05	0.000149681839	0.000199532508	0.000199604034	0.000149571895	4.99E-05	9.97E-05	4.95E-05	9.96E-05	0.000132843852	132.843852
500	0.000947463512	0.000896024704	0.001095330715	0.000997316837	0.001048409939	0.001047015190	0.000950193405	0.000953972339	0.001047205925	0.000899493694	0.000926601888	0.001047194004	0.000988018512	988.0185127
1000	0.001882636547	0.001893222332	0.001845002174	0.001845240593	0.002044379711	0.001967763901	0.001894915104	0.001890587807	0.002444124222	0.001842474937	0.001895308495	0.001919829845	0.001947123806	1947.123806
5000	0.012047529220	0.012049210070	0.012020158770	0.011821603780	0.011765193940	0.011715960500	0.011691100830	0.011951100830	0.012222158910	0.011772000790	0.012268245220	0.011823940280	0.011952141920	11952.14192
10000	0.028333740950	0.027694082260	0.028498766810	0.027625381950	0.027273511890	0.028300940990	0.027628350260	0.027483820920	0.027420997620	0.027085316180	0.027743244170	0.027780544760	0.027739365900	27739.3659
50000	0.2778176546	0.2724867582	0.2674284101	0.2778347731	0.2735579133	0.2681139708	0.2762591362	0.2785429716	0.2808054328	0.2756137848	0.2729730964	0.2737776875	0.2746009658	274600.9658

Valores N QuickSort (Sara)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000142478942	0.000101339817	5.16E-05	0.000245356559	5.19E-05	0.000228810310	4.97E-05	0.000307810306	0.000144183635	5.00E-05	0.000190448761	0.000339353084	0.000158577164	158.577164
500	0.000160074234	0.000660657882	0.001503086090	0.000101625919	0.000840151310	0.000490188598	0.001731479168	0.000825846195	0.000783109664	0.000891983509	0.001608128094	0.000950598716	0.000878967841	878.9678415
1000	0.001196920872	0.002418279648	0.001641106606	0.001003616714	0.001652884483	0.001591134071	0.000787758827	4.99E-05	0.001728737354	0.001614558697	0.000781023502	0.001562988758	0.001260739565	1260.739565
5000	0.010560250280	0.008046925068	0.008097803593	0.008841979504	0.007335007191	0.009434401989	0.011371600630	0.005822384357	0.010671103	0.006586933136	0.007403159142	0.008374750750	0.008545418580	8545.41858
10000	0.017440867420	0.019070470330	0.015852904320	0.018011343480	0.017687654500	0.017484283450	0.016755890850	0.016123795510	0.016796460480	0.016038000580	0.018292884430	0.019298668460	0.017404599980	17404.59998
50000	0.1011550426	0.1049347758	0.1042585135	0.1010623813	0.0989275813 1	0.1018079042	0.1057864189	0.1018840909	0.1007778168	0.0999944925 3	0.1021772027	0.1041742563	0.1022450397	102245.0397

Valores N MergeSort (Sara)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000537788868	0.000490438938	0.000665318965	0.000127315521	0.000448131561	0	0.001009023190	0.000303876400	0	0.000287950038	1.73E-05	1.20E-05	0.000324923793	324.9237935
500	0.001624548435	0.001728391647	0.000665318965	0.000859904289	0.001558482647	0.001600628470	0.001642119884	0.001591396332	0.000839424133	0.000887525081	0.001838660240	0.001626396179	0.001371852557	1371.852557
1000	0.001699972153	0.002324545383	0.002411496639	0.003265321255	0.001634013653	0.003208363056	0.001768195629	0.002503561974	0.002582025528	0.001749670506	0.002725541592	0.002527487278	0.002366682887	2366.682887
5000	0.015720307830	0.015866529940	0.016257011890	0.017730975150	0.016816163060	0.015545601940	0.015528666970	0.014668202400	0.015131678850	0.017024552820	0.015251159670	0.016293191910	0.015977853540	15977.85354
10000	0.035458326340	0.038720190530	0.034484088420	0.036414873600	0.038623595240	0.036983549590	0.039495396610	0.033808767800	0.036952233310	0.037788724900	0.035158681870	0.038489139080	0.036864797270	36864.79727
50000	0.3566276073	0.356997776	0.3481882453	0.3519000173	0.3537207603	0.3490009546	0.3574926019	0.3560885310	0.3539679050	0.3589042068	0.3549618363	0.3595421076	0.3547827125	354782.7125

Valores N QuickSort (Nicole)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000400340557	0	0.000399947166	0.0003709912 3	0	0	0	2.61E-06	0	0	0.000398875 61	7.67E-06	0.000131786863	131.786863
500	0.001255357265	0.001219642162	0.000801110267	0.001616489887	0.001228833199	3.39E-06	0.001632070541	0.001622211933	0.000418031215	0.001618850231	0.001200151443	0.001617598534	0.001186144352	1186.144352
1000	0.003263330460	0.002091956139	0.002423369884	0.002125811577	0.002494215965	0.002047729492	0.002138292789	0.002494323254	0.002488529682	0.001643681526	0.001337206364	0.002032732964	0.002215098341	2215.098341
5000	0.013126504420	0.012084531780	0.012492954730	0.013631308080	0.013654788390	0.014084970950	0.012695026400	0.013195061680	0.012311732770	0.012787961960	0.012573336140	0.012617301940	0.012937958040	12937.95804
10000	0.026295661930	0.028266239170	0.029607820510	0.027409219740	0.027520072460	0.026474034790	0.027243113520	0.028454864030	0.026365435120	0.027702653410	0.026279819010	0.027323341370	0.027411856250	27411.85625
50000	0.1592483044	0.1582477927	0.1599764824	0.1575348139	0.1552066721	0.1573586345	0.1581968665	0.1571631551	0.1567242265	0.1572293282	0.1585639	0.1573791504	0.1577352772	157735.2772

Valores N MergeSort (Nicole)	1	2	3	4	5	6	9	10	11	12	13	14		Resultados
100	0.000400102138	0	0	0	0.000400543212	0.000400042533	0	0.000372433662	0	0.000412249565	0.000400876998	0.000400221347	0.000232205788	232.2057883
500	0.002844953537	0.001273500919	0.002466309071	0.002064776421	0.001248252392	0.002414107323	0.002799952030	0.002400732040	0.002801072598	0.002388358116	0.002412843704	0.002800142765	0.002326250078	2326.250076
1000	0.004613614082	0.004630339146	0.004954588413	0.005327415466	0.004494738579	0.004587531090	0.004801440239	0.004801332951	0.004400384426	0.005201220512	0.004398357868	0.004409968552	0.004717660944	4717.660944
5000	0.030786573890	0.031103777890	0.030540168290	0.030251646040	0.030610883240	0.031871557240	0.030425894260	0.031636142730	0.029629445080	0.031193017960	0.030769801140	0.029991209510	0.030734176440	30734.17644
10000	0.074272704120	0.070507514480	0.072986781600	0.070518183710	0.073301279540	0.071401095390	0.070431244370	0.071220648290	0.071235287190	0.070786476140	0.072443234920	0.070841968060	0.071662201480	71662.20148
50000	0.7312753797	0.7150448322	0.7148213983	0.7170760036	0.7240739465	0.7246505260	0.7195917964	0.7177635908	0.7175537825	0.7270175815	0.7230350375	0.7176594019	0.7207969397	720796.9397

Apéndice B.

Valores FPU Andres	1	2	3	4	5	6	9	10	11	12	13	14	
100	2	1	1	1	2	1	2	1	2	2	1	2	1.5
500	2	3	3	2	2	2	3	3	3	2	2	2	2.416666667
1000	4	4	4	3	3	4	4	4	3	4	3	4	3.666666667
5000	11	12	12	11	12	11	12	12	12	12	11	12	11.66666667
10000	21	21	21	21	21	21	22	21	21	21	22	21	21.16666667
50000	79	78	79	79	80	78	79	79	79	79	79	79	78.91666667

Valores FPU Fer	1	2	3	4	5	6	9	10	11	12	13	14	
100	6	1	0	1	1	0	1	1	1	1	1	2	1.333333333
500	2	2	2	2	2	2	2	1	2	1	2	1	1.75
1000	3	2	2	2	2	3	2	3	3	3	2	3	2.5
5000	9	7	9	9	9	9	10	9	9	9	9	9	8.916666667
10000	15	16	13	16	15	16	15	17	17	17	17	16	15.83333333
50000	58	58	58	60	60	72	65	60	58	59	62	61	60.91666667

Valores FPU Sara	1	2	3	4	5	6	9	10	11	12	13	14	
100	3	2	2	2	1	2	2	3	3	1	2	2	2.083333333
500	4	4	4	5	4	4	4	4	4	4	4	3	4
1000	5	6	6	6	6	5	5	5	5	5	5	5	5.666666667
5000	8	13	12	13	14	9	5	8	14	10	9	14	10.75
10000	25	22	24	22	24	23	22	23	20	25	25	17	22.66666667
50000	87	88	85	89	85	85	91	87	81	84	90	92	87

Valores FPU Nicole	1	2	3	4	5	6	9	10	11	12	13	14	
100	0	0	8	0	0	0	0	8	8	8	0	8	3.333333333
500	0	8	0	0	8	0	8	8	8	8	8	8	5.333333333
1000	8	8	8	8	8	16	8	8	8	8	8	8	8.666666667
5000	24	24	24	24	24	16	24	24	24	16	24	24	22.66666667
10000	40	40	40	40	32	40	40	40	40	40	40	40	39.33333333
50000	153	144	144	144	152	152	152	152	152	144	160	144	149.4166667

Apéndice C.

www.phoronix-test-suite.com

Prueba 1 del Computador A

AMD Ryzen 7 2700X Eight-Core testing with a Gigabyte AX370M-DS3H-CF (F23 BIOS) and NVIDIA GeForce RTX 2070 8GB on Microsoft Windows 10 Home Build 18363 via the Phoronix Test Suite.

Test Systems:

NVIDIA GeForce RTX 2070

Processor: AMD Ryzen 7 2700X Eight-Core @ 3.70GHz (8 Cores / 16 Threads), Motherboard: Gigabyte AX370M-DS3H-CF (F23 BIOS), Memory: 2 x 8192 MB 2666MHz, Disk: 932GB WDC WD10EZEX-00WN4A0 + 224GB ADATA SU650, Graphics: NVIDIA GeForce RTX 2070 8GB, Audio: Sennheiser Main Audio + NVIDIA HD Audio + Sennheiser Communication Audio + NVIDIA Virtual Audio Device (Wave Extensible) (WDM) + Realtek HD Audio, Monitor: GM3CS4, Network: Intel Dual Band Wireless-AC 3168 + Realtek PCIe GbE

OS: Microsoft Windows 10 Home Build 18363, Kernel: 10.0 (x86_64), Display Driver: 472.12 (30.0.14.7212), File-System: NTFS, Screen Resolution: 1920x1080

Processor Notes: CPU Microcode: 0682000800000000
Security Notes: __user pointer sanitization: Disabled + Retpoline: Full + IBPB: Always

	NVIDIA GeForce RTX 2070
GpuTest - Furmark - 800 x 600 - Windowed (Points)	27995
Standard Deviation	1.8%
GpuTest - Furmark - 800 x 600 - Fullscreen (Points)	27042
Standard Deviation	0.2%

Prueba 1 del Computador A

This file was automatically generated via the Phoronix Test Suite benchmarking software on Thursday, 30 September 2021 03:39.

www.phoronix-test-suite.com

Prueba 1 del Computador B

Intel Core i9-10885H testing with a LENOVO 20TK001JUS (N2VET26W 1.11 BIOS) and NVIDIA GeForce GTX 1650 Ti with Max-Q Design + Intel UHD 4GB on Microsoft Windows 10 Pro Build 19042 via the Phoronix Test Suite.

Test Systems:

NVIDIA GeForce GTX 1650 Ti with Max-Q Design

Processor: Intel Core i9-10885H @ 2.40GHz (8 Cores / 16 Threads), Motherboard: LENOVO 20TK001JUS (N2VET26W 1.11 BIOS), Memory: 1 x 32768 MB 2933MHz Samsung M471A4G43AB1-CWE, Disk: 954GB SAMSUNG MZVLB1T0HBLR-000L7, Graphics: NVIDIA GeForce GTX 1650 Ti with Max-Q Design + Intel UHD 4GB, Audio: NVIDIA HD Audio + Intel Display Audio + Realtek HD Audio(SST) + Intel Smart Sound (Intel SST), Monitor: Cannot convert value "True" type "System.Char". Error: " cast from Boolean Char ."At line:1 char:109+ ... $Name = $($_.UserFriendlyName-notmatch 0 | ForEach{[char]$_})-join ...+ ~~~~~~~~ + CategoryInfo : Argument: (:) [] RuntimeException + FullyQualifiedErrorId : CastI, Network: Intel Wi-Fi 6 AX201 160MHz + Lenovo USB

OS: Microsoft Windows 10 Pro Build 19042, Kernel: 10.0 (x86_64), Display Driver: 471.41 (30.0.14.7141), File-System: NTFS, Screen Resolution: 1920x1080

Processor Notes: CPU Microcode: 00000000E2000000

	NVIDIA GeForce GTX 1650 Ti with Max-Q Design
GpuTest - Furmark - 800 x 600 - Windowed (Points)	7744
Standard Deviation	2%
GpuTest - Furmark - 800 x 600 - Fullscreen (Points)	7507
Standard Deviation	1%

Prueba 1 del Computador B

This file was automatically generated via the Phoronix Test Suite benchmarking software on Thursday, 30 September 2021 03:52.

www.phoronix-test-suite.com

Prueba 1 del Computador C

Intel Core i5-8250U testing with a ASUS X510UF (X510UF.303 BIOS) and NVIDIA GeForce MX130 + Intel UHD 620 2GB on Microsoft Windows 10 Home Single Language Build 19043 via the Phoronix Test Suite.

Test Systems:

NVIDIA GeForce MX130

Processor: Intel Core i5-8250U @ 1.80GHz (4 Cores / 8 Threads), Motherboard: ASUS X510UF (X510UF.303 BIOS), Memory: 1 x 8192 MB 2400MHz Samsung M471A1K43CB1-CRC, Disk: 894GB KINGSTON SA400S37960G, Graphics: NVIDIA GeForce MX130 + Intel UHD 620 2GB, Audio: Conexant SmartAudio HD + Sonido Intel para pantallas, Monitor: No se puede convertir el valor "True" al tipo "System.Char". Error: "Conversin no vlida desde Boolean hasta Char ."En lnea: 1 Carcter: 109+ ... $Name = $($_.UserFriendlyName-notmatch 0 | ForEach{[char]$_})-join ...+ ~~~~~~~~ + CategoryInfo : Argument: (:) [] RuntimeException + FullyQualifiedErrorId : CastI, Network: Intel Dual Band Wireless-AC 8265

OS: Microsoft Windows 10 Home Single Language Build 19043, Kernel: 10.0 (x86_64), Display Driver: 451.67 (27.21.14.5167), File-System: NTFS, Screen Resolution: 1366x768

Security Notes: __user pointer sanitization: Disabled + IBPB: Always + IBRS: Enabled + STIBP: Enabled + KPTI Enabled: Yes + PTE Inversion: Yes

	NVIDIA GeForce MX130
FAHBench (Ns/Day)	18.0234
Standard Deviation	0.4%
GpuTest - Furmark - 800 x 600 - Windowed (Points)	1521
Standard Deviation	20.5%
GpuTest - Furmark - 800 x 600 - Fullscreen (Points)	1159
Standard Deviation	11.7%

This file was automatically generated via the Phoronix Test Suite benchmarking software on Monday, 27 September 2021 00:57.

www.phoronix-test-suite.com

Prueba 1 del Computador D

AMD Ryzen 5 3500U testing with a LENOVO LNVNB161216 (AMCN27WWV1.10 BIOS) and AMD Radeon Vega 8 2GB on Microsoft Windows 10 Home Single Language Build 19042 via the Phoronix Test Suite.

Test Systems:

AMD Radeon Vega 8

Processor: AMD Ryzen 5 3500U @ 2.10GHz (4 Cores / 8 Threads), Motherboard: LENOVO LNVNB161216 (AMCN27WWV1.10 BIOS), Memory: 4096 + 8192 2400MHz, Disk: 1863GB WDC WD20SPZX-08UA7, Graphics: AMD Radeon Vega 8 2GB, Audio: Realtek HD Audio + AMD HD Audio Device, Monitor: No se puede convertir el valor "True" al tipo "System.Char". Error: "Conversin no vlida desde Boolean hasta Char ."En lnea: 1 Carcter: 109+ ... $Name = $($_.UserFriendlyName-notmatch 0 | ForEach{[char]$_})-join ...+ ~~~~~~~~ + CategoryInfo : Argument: (:) [] RuntimeException + FullyQualifiedErrorId : CastI, Network: Qualcomm Atheros QCA9377 Wireless

OS: Microsoft Windows 10 Home Single Language Build 19042, Kernel: 10.0 (x86_64), Display Driver: 27.20.11036.3, OpenCL: OpenCL 2.1 AMD-APP (3075.13), File-System: NTFS, Screen Resolution: 1366x768

Processor Notes: CPU Microcode: 0281100800000000
Security Notes: __user pointer sanitization: Disabled + Retpoline: Full + IBPB: Always

	AMD Radeon Vega 8
GpuTest - Furmark - 800 x 600 - Windowed (Points)	1539
GpuTest - Furmark - 800 x 600 - Fullscreen (Points)	1543
Standard Deviation	0.1%

Prueba 1 del Computador D

This file was automatically generated via the Phoronix Test Suite benchmarking software on Tuesday, 28 September 2021 08:46.