**OBSERVACIONES DE LA PRÁCTICA**

Jesed Alejandro Dominguez Piratova Cod 202011992

José Daniel Montero Gutiérrez Cod 202012732

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz | AMD Ryzen 5 2600X Six-Core Processor 3.60 GHz |
| Memoria RAM (GB) | 20 GB | 8.0 GB |
| Sistema Operativo | Win 10 Pro 64- Bits | Windows 10 Pro 64-Bits |

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

# **Maquina 1**

## **Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga (PROBING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0.30 | 1548651,791 | 45505,293 |
| 0.50 | 1485637,899 | 40634,233 |
| 0.80 | 1450449,863 | 39822,937 |

Tabla 2. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 1.

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 2.00 | 1503538,677 | 40549,407 |
| 4.00 | 1441382,955 | 41586,742 |
| 6.00 | 1423742,982 | 42384,519 |

Tabla 3. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 1.

## **Graficas**

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 1.**

* Comparación de memoria y tiempo de ejecución para PROBING y CHAINING

# **Maquina 2**

## **Resultados**

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo PROBING*** | | |
| **Factor de Carga** (PROBING) | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 0.30 | 1439513,626 | 37357,838 |
| 0.50 | 1423754,623 | 35998,065 |
| 0.80 | 1414965,149 | 37077,502 |

Tabla 4. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 2.

|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución [ms]** |
| 2.00 | 1428230,136 | 36767,313 |
| 4.00 | 1423743,441 | 36733,316 |
| 6.00 | 1422246,084 | 36969,339 |

Tabla 5. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 2.

## **Graficas**

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 2.**

* Comparación de memoria y tiempo de ejecución para PROBING y CHAINING

# **Preguntas de análisis**

1. ¿Por qué en la función **getTime()** se utiliza **time.perf\_counter()** en ves de la previamente conocida **time.process\_time()**?

Debido a que la función *time.perf\_counter()* devuelve el valor del tiempo transcurrido realizando un proceso de rendimiento, **incluyendo el tiempo que transcurre mientras la máquina está descansando**. Este tiempo que no incluye la otra función mencionada es necesario para poder saber verdaderamente la duración de la ejecución de la tarea a estudiar.

Se puede comparar *perf\_counter()* a usar un cronómetro, solo tiene el tiempo de inicio y el tiempo final de una tarea, mientras que *process\_time()* no incluye los tiempos en que la computadora descansa, debido a que (a menos que la computadora se dedique a ejecutar solamente el código a probar) la computadora está realizando muchos otros procesos en el fondo.

1. ¿Por qué son importantes las funciones **start()** y **stop()** de la librería **tracemalloc**?

La función *start()* empieza a rastrear los bloques de memoria usados por la función ejecutada, por lo que no incluye los bloques de memoria usados por otras funciones, y al usar *stop()* tenemos un intervalo cerrado, por decirlo de cierta manera, que nos permite conocer los bloques de memoria usados solo durante la ejecución de la función ejecutada.

1. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de videos?

El cambio más notable que se percibe al cambiar el factor de carga máximo es que la velocidad en la que se carga el catálogo es más eficiente según se le vaya incrementando el factor de carga máximo sin importar el sistema de colisiones utilizado.

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de videos?

En términos de memoria, parece ser que entre más grande es el factor de carga esta se va reduciendo el consumo que genere la memoria sin importar el esquema de colisiones utilizado.

1. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

Según los datos obtenidos, podemos intuir que el esquema de colisiones *CHAINING* consume menos tiempo base que el esquema *PROBING*. Debido a que viendo las tablas, se puede concluir que con el factor de carga inicial de cada uno se nota una diferencia en cuento al tiempo de ejecución .

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

El consumo de memoria (en la máquina 2) al usar el esquema de colisiones *PROBING* no es tan constante como si usáramos *CHAINING*, pues la diferencia de consumo de memoria entre el menor y mayor factor de carga del *PROBING* es de 25.000 kBs, mientras que al comparar esta misma estadística con la de el esquema de *CHAINING,* vemos que la diferencia es de 6.000 kBs. Pero sobre todo, al comparar las dos máquinas, Podemos ver que el esquema *CHAINING* parece consumer constantemente menos memoria que el *PROBING.*