**OBSERVACIONES DEL LA PRACTICA**

Estudiante 1 Felipe Rueda Cod 202010903

Estudiante 2 Julian Parra Cod 202013033

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel® Core™ i5-1035G7 CPU @1.20GHz 1.50GHz | Intel® Core™ i7-10870H CPU @2.20GHz 2.21GHz |
| Memoria RAM (GB) | 8.00 GB | 8.00 GB |
| Sistema Operativo | Windows 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 | 1726199.061 | 41729.395 |
| 0.50 | 1726199.295 | 42468.356 |
| 0.80 | 1726199.764 | 37480.750 |

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 | 1813514.709 | 43784.869 |
| 4.00 | 1813514.709 | 39784.584 |
| 6.00 | 1813514.709 | 39694.953 |

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 | 1725189.435 | 30620.468 |
| 0.50 | 1726189.739 | 28316.080 |
| 0.80 | 1726189.591 | 25150.821 |

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 | 1755294.495 | 27430.372 |
| 4.00 | 1755294.651 | 26488.643 |
| 6.00 | 1755294.502 | 25660.798 |

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 vez de la previamente conocida **time.process\_time()**?

Aunque los dos tratan con el tiempo se usa time.perf\_counter() ya que esta cuenta el tiempo total, mientras time.process\_time() solo cuenta el tiempo parcial, No incluye el tiempo transcurrido durante el sueño.

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

Las funciones start y stop son importantes pues con start Python sabe que debe empezar a inicializar el proceso para medir la memoria que sea utilizada en la función que indique el usuario. Igualmente, con stop Python sabe que debe terminar el proceso de medir la memoria que sea utilizad por la misma función.

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

Se puede evidenciar que a medida que el factor de carga aumenta, el tiempo de ejecución disminuye. Por lo que, es inversamente proporcional.

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

Se puede evidenciar que sin importar que el factor de carga aumente o disminuya el consumo de energía siempre se mantiene constante, por lo que, durante los cambios del factor de carga el consumo de memoria siempre es el mismo.

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.

Se puede evidenciar que el tiempo de ejecución es menor utilizando el esquema de colisiones llamado *Separate Chaining* en ambas maquinas en comparación con *Lineal Probing,* pues se evidencia que el mayor tiempo de *Separate Chaining* es de 27430.372 en cambio el tiempo mayor de *Lineal Probing* es 30620.468 (estoy comprando el peor caso y esto aplica cuando el factor de carga es 0.5 el cual es el recomendado por el curso)*,* A pesar de que con cierto factor de carga se intercambia los papeles pues en otro factor de carga *Lineal Probing (25150.821 ms)* tiene menor tiempo de ejecución que *Separate Chaining (25660.798 ms).*Esto se puede evidenciar teóricamente pues en el peor caso del esquema de colisiones *Lineal Probing* tiene una complejidad de O(n), pues cuando hay una colisión tiene que buscar una posición después si esa posición está vacía y así hasta que encuentre una posición vacía para ser guardada ahí.

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.

Se puede observar que en cuanto el consumo de memoria el esquema de colisiones que hace más consumo de memoria es el *Separate Chaining* en comparación con el *Lineal Probing,* pues en promedio el consumo de *Separate Chaining* es de 1755294 Kb (en caso de la maquina 2) y en promedio de *Lineal Probing* es de 1725189 Kb. Esto es debido a que en Separte Chaining se crea un bucket o en una lista, esto quiere decir que el consumo de memoria es mayor pues a parte de crear lla tabla hash se esta creando una lista o bucket. En cambio, Lineal Probing no crea una lista pues si la posición esta ocupada busca la siguiente posición que no este ocupada.