**OBSERVACIONES DEL LA PRACTICA**

Estudiante 1:Juan Felipe García 202014961 jf.garciam1

Estudiante 2: Santiago Rodríguez 202020476 s.rodriguez64

|  |  |  |
| --- | --- | --- |
|  | Máquina 1 | Máquina 2 |
| Procesadores | Intel(R) Core (TM) i5-10300h CPU @2.50GHZ 2.50GHZ | Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50GHz |
| Memoria RAM (GB) | 8Gb | 12 GB |
| Sistema Operativo | Windows 10 64-bits | Windows 10 Home Single Language 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 |  |  |
| 0.50 |  |  |
| 0.80 |  |  |

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 |  |  |
| 4.00 |  |  |
| 6.00 |  |  |

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 | 1.327.229,729 | 27.501,333 |
| 0.50 | 1.327.216,557 | 25.872,620 |
| 0.80 | 1.327.203,328 | 25.009,815 |

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 | 1.327.249,331 | 26.056,817 |
| 4.00 | 1.327.249,170 | 28.826,078 |
| 6.00 | 1.327.248,876 | 29.637,399 |

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()**?

time.process\_time() cuenta solo el tiempo que se demora la operación en ejecutar, time.perf\_counter() mide el tiempo real que se pasa. Como aquí se quiere medir el tiempo que pasa desde que se llama la función hasta que entrega el resultado es lógico usar time.perf\_counter().

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

Start() habilita que se pueda medir la memoria y stop() cierra esta posibilidad.

De este modo se usan para empezar a contar la memoria utilizada en los procesos y detenerla. Lo que sirve para conocer la información y hacer otros procesos como delta memoria.

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

En linear probing mientras más pequeño el factor, más eficiente es porque tiene que revisar menos colisiones. En separate chaining aumenta el tiempo al mismo tiempo que aumenta el factor de carga.

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

En linear probing mientras más pequeño el factor, es mayor el consumo de memoria, pues se necesitan tablas más grandes. En separate chaining el consumo en memoria se mantiene casi igual, aumenta muy poco

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.

En general probing es más rapido que chaining pues todo se encuentra en una sola estructura. Asi todos los datos están más cerca en la memoria y es más sencillo acceder a ellos.

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.

Probing eusa menos memoria que chaining pues chaining contiene subestructuras que son listas por dentro que aumentan el uso de memoria comparado con probing que usa solo una estructura.