**OBSERVACIONES DEL LA PRACTICA**

Estudiante 1 Cod XXXX

Estudiante 2 Cod XXXX

Estudiante 3 Cod XXXX

|  |  |  |  |
| --- | --- | --- | --- |
|  | Máquina 1 | Máquina 2 | Máquina 3 |
| Procesadores | AMD Ryzen 5 5500U with Radeon Graphics, 2100 Mhz, 6 procesadores principales, 12 procesadores lógicos |  |  |
| Memoria RAM (GB) | 8.00 GB |  |  |
| Sistema Operativo | Microsoft Windows 11 Home Single Language |  |  |

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 Real @LP [ms]** |
| 0.1 | 60663.401 | 4865.086 |
| 0.5 | 50510.780 | 4488.523 |
| 0.7 | 49806.601 | 4959.302 |
| 0.9 | 49419.565 | 7154.891 |

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.

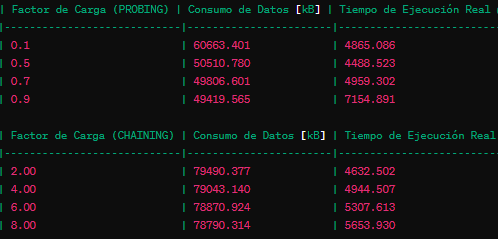
|  |  |  |
| --- | --- | --- |
| ***Carga de Catálogo CHAINING*** | | |
| **Factor de Carga (CHAINING)** | **Consumo de Datos [kB]** | **Tiempo de Ejecución Real @SC [ms]** |
| 2.00 | 79490.377 | 4632,502 |
| 4.00 | 79043.140 | 4944.507 |
| 6.00 | 78870.924 | 5307.613 |
| 8.00 | 78790.314 | 5653.930 |

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



El análisis de la tabla proporcionada muestra algunos patrones interesantes en términos de consumo de datos y tiempo de ejecución para los métodos de Linear probing(PROBING) y Separate Chaining (CHAINING) con diferentes factores de carga:

1. Consumo de datos:

* En general, el consumo de datos parece ser más alto para el método CHAINING en comparación con PROBING para factores de carga similares.
* A medida que aumenta el factor de carga en CHAINING, el consumo de datos tiende a aumentar, lo que indica que este método puede requerir más espacio de memoria a medida que se incrementa la carga.

1. Tiempo de ejecución:

* En términos de tiempo de ejecución, PROBING parece ser más rápido en general en comparación con CHAINING para los factores de carga dados.
* Para el método CHAINING, se observa un aumento en el tiempo de ejecución a medida que aumenta el factor de carga, lo que sugiere que el método puede volverse menos eficiente en términos de tiempo a cargas más altas.

# **Preguntas de análisis**

1. ¿Por qué en la función **getTime()** se utiliza **time.perf\_counter()** en vez de otras funciones como **time.process\_time()**?

* En la función “getTime()”, se utiliza porque “time.perf\_counter()” proporciona una medida más precisa del tiempo de procesamiento en milisegundos, especialmente en entornos donde la precisión del tiempo es crítica, como en mediciones de rendimiento. “time.perf\_counter()” devuelve una medida de tiempo que es independiente del tiempo del sistema y que tiene una mayor resolución que “time.process\_time()”.

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

* Las funciones “start()” y “stop()” son importantes porque permiten realizar un seguimiento detallado del uso de la memoria. La función start(), inicia el rastreo de asignaciones de memoria, lo que significa que a partir de este punto, “tracemalloc” comenzará a registrar información sobre las asignaciones de memoria realizadas por el programa. La función stop(), detiene el rastreo de asignaciones de memoria, después de llamar a stop(), “tracemalloc” deja de registrar la información sobre las asignaciones de memoria.

1. ¿Por qué no se puede medir paralelamente el **uso de memoria** y el **tiempo de ejecución** de las operaciones?

* No es posible medir de manera simultánea y directa el uso de memoria y el tiempo de ejecución de las operaciones debido a sus diferentes métricas y unidades de medida, así como el impacto en el rendimiento y la posible interferencia entre las mediciones.

1. Teniendo en cuenta cada uno de los requerimientos del reto ¿Cuántos índices implementaría en el Reto? y ¿Por qué?

* Para el reto sería apropiado implementar varios índices para optimizar las consultas como los siguientes:

1. Índice por país: Un índice que permita buscar ofertas de trabajo por país. Esto facilitará las consultas relacionadas con países específicos, como en los requerimientos 1,4,5 y 8.
2. Índice por nivel de experticia: Otro índice para buscar ofertas de tabajo según el nivel de experticia. Esto será útil para el requerimiento 1 y posiblemente para otros que requieran filtrar por nivel de experiencia.
3. Índice por empresa: Un índice que vincule las ofertas de trabajo con la empresa que las publicó, esto serviría para el requerimiento 3, donde se busca consultar ofertas por empresa durante un periodo de tiempo específico.
4. Índice por ciudad: Un índice por ciudad sería útil para para consultas que involucren ciudades, como el requerimiento 5.
5. Índice por fecha de publicación: Un índice que ordene las ofertas por fecha de publicación será útil para el requerimiento 3 y cualquier otra consulta que necesite filtrar por periodos de tiempo.
6. Índice para clasificación: si se requiere clasificar ciudades o países según el número de ofertas de trabajo o experticia, se podría implementar un índice adicional que facilite esta clasificación sin tener que realizar cálculos complejos en tiempo real.
7. Según los índices propuestos ¿en qué caso usaría **Linear Probing** o **Separate Chaining** en estos índices? y ¿Por qué?
8. Índice por país y ciudad: Linear Probing, dado que se espera un número limitado de colisiones y un tamaño de índice relativamente pequeño para los países y ciudades, Linear Probing podría ser el mas adecuado. Proporciona una implementación mas simple y eficiente en este contexto.
9. Índice por nivel de experticia y fecha de publicación: Separate Chaining, considerando la diversidad de niveles de experticia y la posibilidad de muchas ofertas de trabajo publicadas en diferentes fechas, es probable que haya colisiones frecuentes. Por lo tanto, Separate Chaining es el más apropiado para este índice.
10. Índice por empresa y clasificación: Linear Probing, dado que el número de empresas y clasificaciones puede no ser tan grande como otros índices, y las colisiones podrían ser menos frecuentes, Linear Probing sería una elección adecuada aquí debido a su simplicidad y menor uso de memoria.
11. Dado el número de elementos de los archivos del reto (large), ¿Cuál sería el factor de carga para estos índices según su mecanismo de colisión?
12. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar las ofertas de trabajo?
13. Factor de carga en PROBING:

* Para PROBING, se observa que a medida que el factor de carga aumenta de 0.1 a 0.9, el tiempo de ejecución real también aumenta significativamente, pasando de alrededor de 4865 ms a 7154 ms. Esto indica que cargar las ofertas de trabajo con un factor de carga más alto en PROBING resulta en un tiempo de ejecución más largo, posiblemente debido a un mayor numero de colisiones y reubicaciones.

1. Factor de carga en CHAINING:

* En el caso de CHAINING, también se observa un aumento en el tiempo de ejecución a medida que el factor de carga aumenta. Por, para un factor de carga de 2.00, el tiempo de ejecución es de alrededor de 4632 ms, mientras que para un factor de carga de 8.00, el tiempo de ejecución aumenta a 5653 ms. Esto sugiere que cargar las ofertas de trabajo con un factor de carga más alto en CHAINING también resulta en un tiempo de ejecución más largo, posiblemente debido al aumento en la longitud de listas enlazadas y las operaciones asociadas.

1. ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar las ofertas de trabajo?
2. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

1. Factor de carga en probing:

* En PROBING, se observa que el consumo de datos en kilobytes(kb) disminuye a medida que aumenta el factor de carga máximo. Por ejemplo, para un factor de carga de 0.1, el consumo es de aproximadamente 60663.401 kb, mientras que para un factor de carga de 0.9, el consumo disminuye a alrededor de 49419.565 kb. Esta tendencia de disminución en el consumo de memoria podría deberse a la forma en que se manejan las colisiones en el método de probing, lo que permite una distribución más eficiente de las claves.

2. Factor de carga en CHAINING:

* En CHAINING, el consumo de datos en kb tiende a aumentar a medida que aumenta el factor de carga máximo. Por ejemplo, para un factor de carga de 2.00, el consumo es de alrededor de 79490.377 kb, mientras que para un factor de carga de 8.00, el consumo aumenta a 78790.314 kb. Esta tendencia de aumento en el consumo de memoria podría estar relacionada con el crecimiento de las listas enlazadas y el numero de elementos almacenados en cada lista a medida que se incrementa la carga.

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.
2. Linear Probing:

* El consumo de memoria puede ser menor en comparación con chaining para un mismo conjunto de datos y factor de carga. Esto se debe a que en probing, las claves con colisiones se almacenan directamente en la tabla hash, evitando la necesidad de estructuras de datos adicionales como las listas enlazadas.

1. Separate Chaining:

* El consumo de memoria tiende a ser mayor debido a la necesidad de mantener listas enlazadas para manejar colisiones. Cada entrada de la tabla hash en chaining debe mantener una lista que pueda contener múltiples elementos, lo que puede aumentar el uso de memoria, especialmente cuando hay muchas colisiones.

1. ¿Qué configuración de ideal ADT Map escogería para el **índice de ofertas de trabajo por “id”**?, especifique el mecanismo de colisión, el factor de carga y el numero inicial de elementos.
2. Mecanismos de colisión: Probing puede ser más eficiente en términos de uso de memoria directa en la tabla de hash, mientras que chaining puede ofrecer un mejor manejo de colisiones y rendimiento en ciertos casos.
3. Factor de carga: El factor de carga ideal puede variar dependiendo de la implementación y la capacidad de manejar colisiones de la estructura de datos. Un factor de carga cercano a 1.0 puede ser adecuado para mantener un buen equilibrio entre el uso de memoria y el rendimiento.
4. Numero inicial de elementos: El número inicial de elementos en la tabla hash puede determinarse según la cantidad de datos esperados y la distribución de las claves. Es importante tener suficiente espacio inicial para evitar rehashing frecuente y mantener un rendimiento óptimo.