






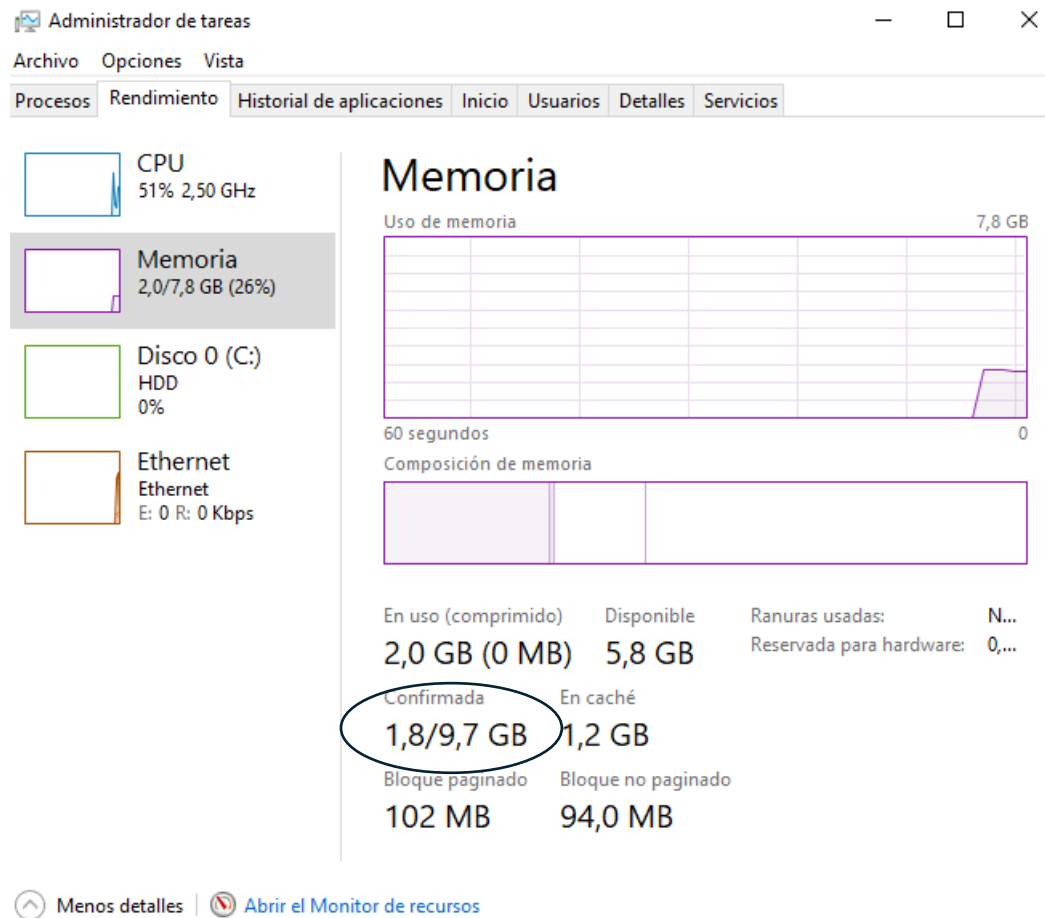


LABORATORIO 2: GESTIÓN DE MEMORIA

Para las siguientes pruebas se utilizó las siguientes especificaciones

	General
Nombre:	windowslab
Sistema operativo:	Windows 10 (64-bit)
	Sistema
Memoria base:	8000 MB
Procesadores:	2
Orden de arranque:	Disquete, Óptica, Disco duro
Aceleración:	Paginación anidada, Paravirtualización Hyper-V
	Pantalla
Memoria de vídeo:	128 MB
Controlador gráfico:	VBoxSVGA
Servidor de escritorio remoto:	Inhabilitado
Grabación:	Inhabilitado
	Almacenamiento
Controlador:	SATA
Puerto SATA 0:	windowslab.vdi (Normal, 50,00 GB)
Puerto SATA 1:	[Unidad óptica] Vacío
	Audio
Controlador de anfitrión:	Predeterminado
Controlador:	Audio Intel HD
	Red
Adaptador 1:	Intel PRO/1000 MT Desktop (NAT)
	USB
Controlador USB:	xHCI
Filtros de dispositivos:	1 (1 activo)

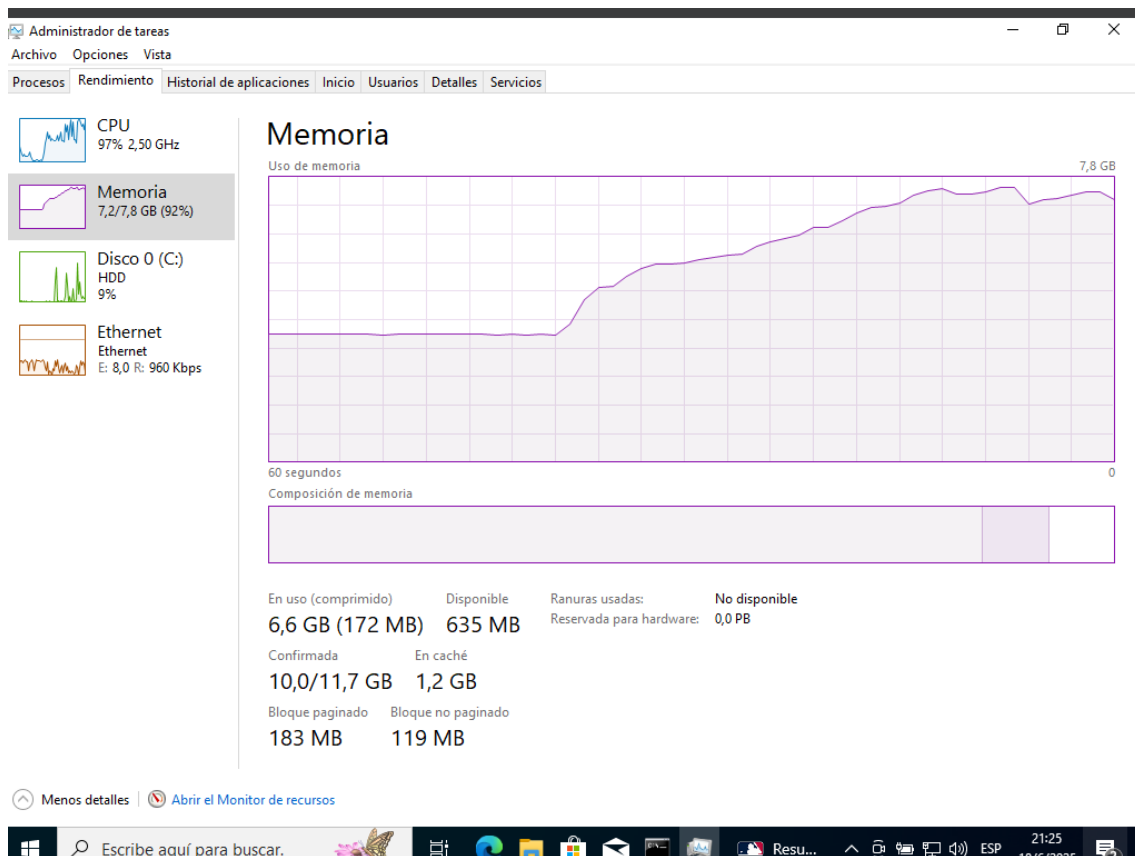
Estados de la memoria virtual y física



Podemos observar que el uso de la memoria es físico y es estable, no es alta a comparación del tamaño de RAM que tiene el equipo, que es de 8(gb)

Para poder llenar la memoria utilice además de los programas y aplicaciones abiertas, un programa en python que, llenaba la memoria ram disponible que era

```
C:\Windows\system32\cmd.exe - python memoria.py
Usado: 5.66 GB
Usado: 5.76 GB
Usado: 5.86 GB
Usado: 5.96 GB
Usado: 6.05 GB
Usado: 6.15 GB
Usado: 6.25 GB
Usado: 6.35 GB
Usado: 6.45 GB
Usado: 6.54 GB
Usado: 6.64 GB
Usado: 6.74 GB
Usado: 6.84 GB
Usado: 6.93 GB
Usado: 7.03 GB
Usado: 7.13 GB
Usado: 7.23 GB
Usado: 7.32 GB
Usado: 7.42 GB
Usado: 7.52 GB
Usado: 7.62 GB
Usado: 7.71 GB
Usado: 7.81 GB
Usado: 7.91 GB
Usado: 8.01 GB
Presiona Enter para liberar memoria...
```



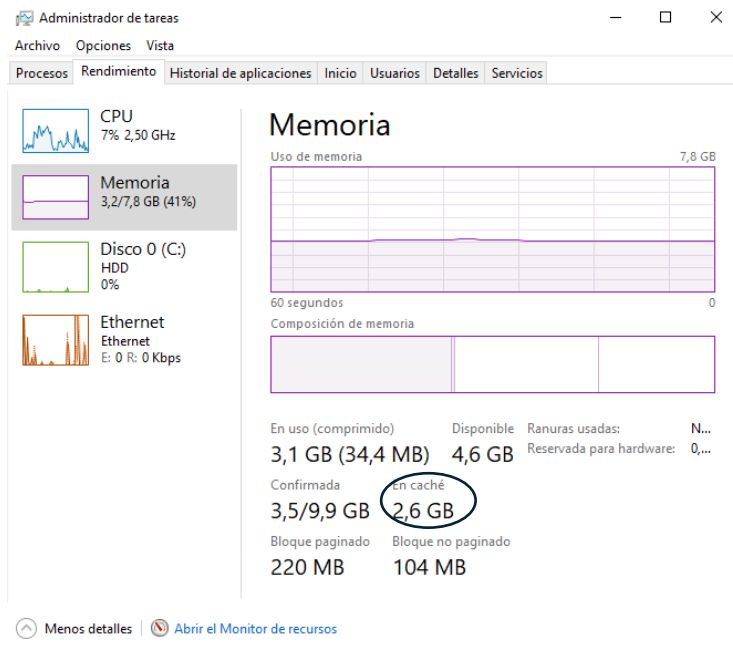
Observe el aumento considerable del uso de la memoria, paso el pico de los 8gb, y ya empezó a trabajar la memoria virtual del equipo.

Cuando la RAM física se llena (92% de uso).

Para que el sistema no se colapse cuando los programas piden más memoria de la disponible.

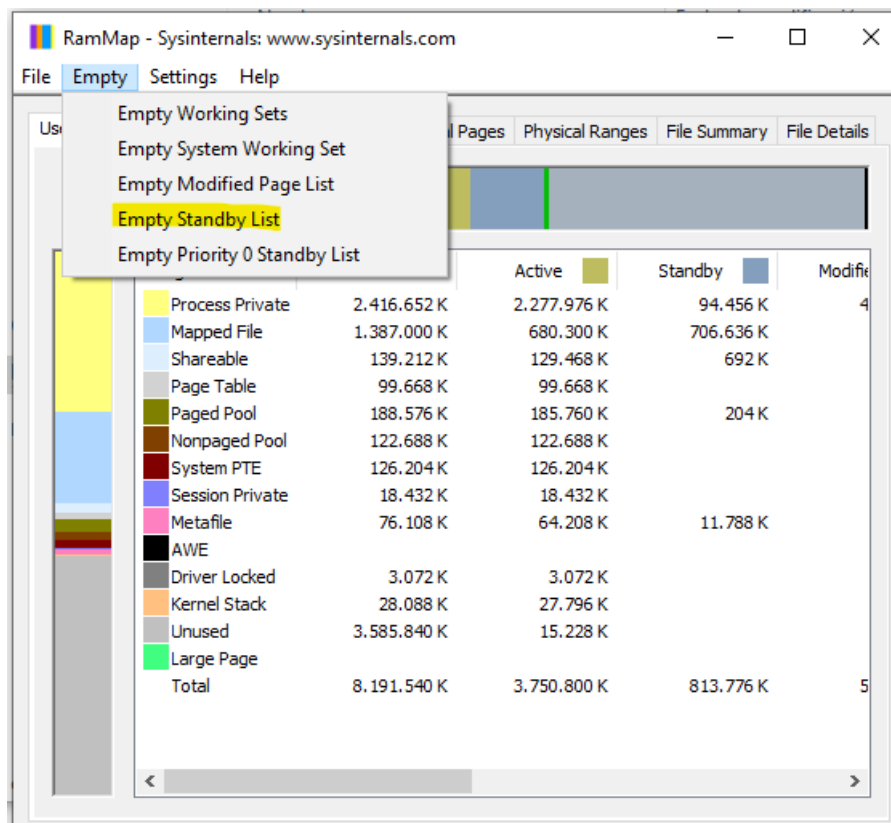
- La máquina tenía 7.8 GB de RAM, pero ya usaba 7.2 GB (92%).
- Como quedaban solo 635 MB libres, Windows activó memoria virtual, alcanzando 10,0/11,7 (gb), según la imagen
- Todo funciona, pero más lento

Cache y rendimiento

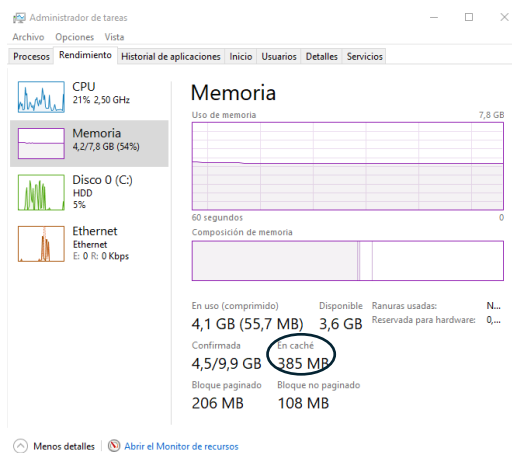


Vemos que el cache en uso es extremadamente alto, el cache es un almacenamiento temporal que permite un acceso rápido a datos que ha sido procesados recientemente. La memoria cache es buena cuando se siguen utilizando la información para ese proceso, si algún programa está mal diseñado, llena de cache con datos inútiles.

En este caso el caché son datos útiles que Windows guardó para agilizar procesos (como el script de Python).



Descargue la herramienta RamMap, para poder eliminar el cache en mi sistema, viendo la imagen en empty standby list, le damos a esta opción, haciendo que los archivos que el sistema guardo temporalmente, y sean incensarios, sean eliminados.



Visualizando, después de haber hecho el paso anterior, vemos que la memoria cache a disminuido, ¿y porque no ha quedado en 0?, hay archivos que no son basura ya que, si el sistema necesita RAM, lo sobrescribe automáticamente.

```
Símbolo del sistema - python cachetest2.py
Microsoft Windows [Versión 10.0.19045.3803]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Thiago>cd desktop

C:\Users\Thiago\Desktop>python cachetest2.py
Inicio prueba de caché y rendimiento
Acceso secuencial: 1.1331 segundos
Acceso aleatorio: 4.7659 segundos
Fin prueba de caché y rendimiento
Presiona Enter para cerrar...
```

Utilice un programa que mide los diferentes accesos que tiene la memoria cache, con referencia al tiempo que tarda la primera imagen nos muestra

- Acceso secuencial es el que más se usa y es rápido, porque la caché del CPU predice y precarga los datos siguientes
- Acceso aleatorio es lento y evitable, porque la caché no puede adivinar el próximo dato y tiene que buscar en la RAM

```
C:\Windows\py.exe
Inicio prueba de caché y rendimiento
Acceso secuencial: 1.9893 segundos
Acceso aleatorio: 9.2440 segundos
Fin prueba de caché y rendimiento
Presiona Enter para cerrar...
```

Reiniciamos el sistema y volvemos a ejecutar el programa

Análisis de resultados tras reinicio:

1. **Acceso secuencial (1.1s → 1.9s)**
 - Leve aumento debido a:
 - Menos caché disponible (L1/L2) por competencia con otros procesos.
 - La caché aún ayuda, pero con menor eficiencia.

2. Acceso aleatorio (4.7s → 9.2s)

- Doble de lento por:
 - Fallos de caché masivos: La naturaleza impredecible del acceso aleatorio.
 - Posible uso de disco: Si la RAM estaba cerca del 92% de uso (como en tus capturas), el sistema recurrió a memoria virtual (swap en disco).
- Tras reinicio, aunque la caché está "limpia", otros procesos del sistema compiten por recursos.
- El acceso aleatorio no se beneficia de la caché, y con RAM limitada.

Codigos de los programas

Cachetest2.py

```
import time
```

```
import numpy as np
```

```
print("Inicio prueba de caché y rendimiento")
```

```
size = 10**7 # Tamaño grande para afectar caché
```

```
array = np.arange(size)
```

```
start = time.time()
```

```
# Acceso secuencial: buen uso de caché
```

```
sum_seq = 0
```

```
for i in range(size):
```

```
    sum_seq += array[i]
```

```
end_seq = time.time()
```

```
print(f"Acceso secuencial: {end_seq - start:.4f} segundos")
```

```

start = time.time()

import random

sum_rand = 0

for _ in range(size):
    sum_rand += array[random.randint(0, size-1)]

end_rand = time.time()

print(f'Acceso aleatorio: {end_rand - start:.4f} segundos')

print("Fin prueba de cache y rendimiento")

input("Presiona Enter para cerrar...")

```

Este código es una demostración práctica de cómo afecta la caché del procesador al rendimiento, comparando dos tipos de acceso a memoria: secuencial (rápido) vs aleatorio (lento). Importante descargar la librería numpy para el código, numpy es una biblioteca fundamental para cálculo numérico en Python.

Time, usa el reloj del sistema, para comparar velocidades de código en la prueba del cache

Conclusión General del Laboratorio

Con este Laboratorio, podemos concluir del uso de la memoria virtual y física del sistema, y como afecta al comportamiento del sistema, cuando sobrepasa los límites de la memoria RAM.

Además, Este lab demostró que la caché del CPU es clave para el rendimiento, donde el acceso secuencial a memoria (1-2 segundos) es ultra rápido porque el CPU precarga dato, mientras que el acceso aleatorio (4-9 segundos), obligando al sistema a buscar en la RAM (o peor, en el disco, si la RAM está llena). Usar numpy y medir con time.time() fue clave para poder sacar los datos.