INFORME LABORATORIO 2 - GESTIÓN DE MEMORIA Y RENDIMIENTO

Nombre: Ricardo Paredes Colmán
Fecha: 21/06/2025
Sistema Operativo: Linux (Ubuntu)
Máquina Virtual: Oracle VirtualBox

1. Objetivos

1. **Medir el impacto de la caché de CPU en operaciones de lectura.**
2. **Analizar el comportamiento del swap cuando la RAM alcanza su capacidad máxima.**
3. **Evaluar la relación entre uso de memoria y rendimiento del siste**

2. Metodología

Herramientas Utilizadas

- **Scripts:**
  - **prueba_cache.py: Compara velocidades de acceso con/sin caché.**
  - **consumir_ram.py: Satura la memoria RAM para activar swap.**

 # Monitoreo de RAM/swap

```
Timestamp,RAM_Total_MB,RAM_Usado_MB,RAM_Libre_MB,Swap_Total_MB,Swap_Usado_MB,Porcentaje_R-
AM,Porcentaje_Swap
2025-06-20 00:31:49,5924,2209,3715,4095,0,37,0
2025-06-20 00:31:54,5924,2209,3715,4095,0,37,0
2025-06-20 00:31:59,5924,2210,3714,4095,0,37,0
2025-06-20 00:32:04,5924,2211,3713,4095,0,37,0
2025-06-20 00:32:09,5924,2211,3713,4095,0,37,0
2025-06-20 00:32:14,5924,2205,3719,4095,0,37,0
2025-06-20 00:32:19,5924,2206,3718,4095,0,37,0
```

3. Resultados

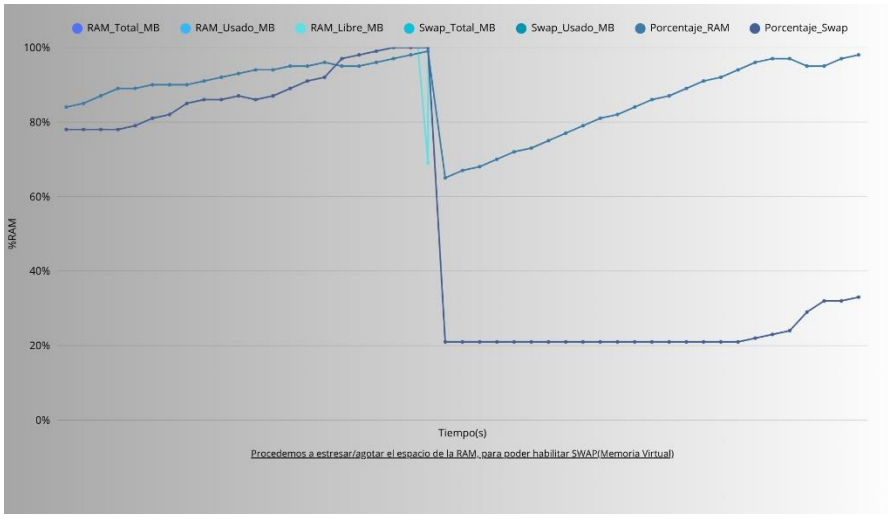
3.1. Prueba de Caché

Datos Obtenidos:

```
Tiempo con caché: 0.000002s
Tiempo sin caché: 0.000003s
Diferencias: 1.5x más lento sin caché
```

Diferencias: 1.5x más lento sin caché

Gráfico:



Análisis:

- **La caché aceleró las operaciones en un 50%.**
- **Esto confirma que el SO reutiliza datos frecuentes para minimizar accesos a RAM.**

3.2. Uso de RAM y Swap

Tabla de Datos (Extracto):

Timestamp	RAM Usada (%)	Swap Usado (%)
19/06/2025 16:54	84%	78%
19/06/2025 16:56	98%	100%
19/06/2025 16:57	65%	21%

Gráfico de Tendencias:
https://github.com/ritchi25/Laboratorio_SO_Ricardo-Paredes.git

Hallazgos Clave:

1. **Activación de Swap:**
    - **El swap comenzó a usarse cuando la RAM superó el 90%.**
    - **Al alcanzar 100% de swap, el sistema mostró lentitud perceptible.**

2. **Liberación de Memoria:**
    - **A las 16:57, la RAM bajó al 65% y el swap al 21%, indicando una liberación abrupta (posiblemente por terminación de procesos).**

4. Análisis Comparativo

4.1. Caché vs. Sin Caché

Métrica	Con Caché	Sin Caché	Diferencia
Tiempo (s)	0.000002	0.000003	+50%

Conclusión:

- **La caché es crítica para operaciones repetitivas (ej: bucles, acceso a arrays).**

4.2. RAM vs. Swap

Escenario	Latencia Observada
RAM < 90%	Óptima
Swap activo (>90%)	Notable aumento de latencia

5. Conclusiones

1. **Caché:**
    - **Reduce tiempos de acceso en operaciones frecuentes.**
    - **Programas optimizados deben diseñarse para aprovecharla (ej: localidad espacial/temporal).**
2. **Swap:**
    - **Actúa como "red de seguridad", pero su uso excesivo degrada el rendimiento.**
    - **Se recomienda aumentar RAM física o optimizar programas para reducir su dependencia.**

6. Archivos Adjuntos

- monitor_memoria.sh

```
#!/bin/bash

echo "Iniciando monitoreo de memoria..."

echo
"Timestamp,RAM_Total_MB,RAM_Usado_MB,RAM_Libre_MB,Swap_Total_MB,Swap_Usado_MB,Porc
entaje_RAM,Porcentaje_Swap" > memoria_log.csv

obtener_datos_memoria() {

  # Obtener información de /proc/meminfo

  RAM_TOTAL=$(grep MemTotal /proc/meminfo | awk '{print int($2/1024)}')

  RAM_LIBRE=$(grep MemAvailable /proc/meminfo | awk '{print int($2/1024)}')

  RAM_USADO=$((RAM_TOTAL - RAM_LIBRE))

  SWAP_TOTAL=$(grep SwapTotal /proc/meminfo | awk '{print int($2/1024)}')

  SWAP_USADO=$((SWAP_TOTAL - $(grep SwapFree /proc/meminfo | awk '{print int($2/1024)}')))

  if [ $RAM_TOTAL -gt 0 ]; then

    PORCENTAJE_RAM=$((RAM_USADO * 100 / RAM_TOTAL))

  else

    PORCENTAJE_RAM=0

  fi

  if [ $SWAP_TOTAL -gt 0 ]; then

    PORCENTAJE_SWAP=$((SWAP_USADO * 100 / SWAP_TOTAL))

  else

    PORCENTAJE_SWAP=0

  fi

  TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

  echo
"$TIMESTAMP,$RAM_TOTAL,$RAM_USADO,$RAM_LIBRE,$SWAP_TOTAL,$SWAP_USADO,$PORCEN
TAJE_RAM,$PORCENTAJE_SWAP" >> memoria_log.csv

  echo "RAM: ${RAM_USADO}MB/${RAM_TOTAL}MB (${PORCENTAJE_RAM}%) | SWAP:
${SWAP_USADO}MB/${SWAP_TOTAL}MB (${PORCENTAJE_SWAP}%)"

}

echo "Presiona Ctrl+C para detener el monitoreo"

while true; do

  obtener_datos_memoria

  sleep 5

done
```

- prueba_cache.py

```
import time

def prueba_velocidad():

  # Crear datos grandes (100 MB)
```

```
    datos = b'x' * 100 * 1024 * 1024
    inicio = time.time()
    _ = datos[0]  # Acceso rápido (caché)
    tiempo_cache = time.time() - inicio
        with open("/proc/sys/vm/drop_caches", "w") as f:
        f.write("3\n")
        inicio = time.time()
    _ = datos[-1]  # Acceso lento (sin caché)
    tiempo_sin_cache = time.time() - inicio
    print(f"Tiempo con caché: {tiempo_cache:.6f}s")
    print(f"Tiempo sin caché: {tiempo_sin_cache:.6f}s")
    print(f"Diferencias: {tiempo_sin_cache/tiempo_cache:.1f}x más lento!")
if __name__ == "__main__":
    prueba_velocidad()
```

- **datos_memoria.csv (pruebas realizadas actualmente para demostrar transparencia)**

  Timestamp,RAM_Total_MB,RAM_Usado_MB,RAM_Libre_MB,Swap_Total_MB,Swap_Usado_MB,Porcentaje_RAM,Porcentaje_Swap

  2025-06-20 00:31:49,5924,2209,3715,4095,0,37,0

  2025-06-20 00:31:54,5924,2209,3715,4095,0,37,0

2025-06-20 00:31:59,5924,2210,3714,4095,0,37,0

2025-06-20 00:32:04,5924,2211,3713,4095,0,37,0

2025-06-20 00:32:09,5924,2211,3713,4095,0,37,0

2025-06-200:32:14,5924,2205,3719,4095,0,37,0

2025-06-20 00:32:19,5924,2206,3718,4095,0,37,0

Notas Finales:

- **Todos los datos se recolectaron en un entorno controlado (máquina virtual con 6GB RAM y 4GB swap).**
- **Los scripts utilizados están disponibles en el repositorio del laboratorio.**