TALLER 3

GESTIÓN DE RECUROS
SISTEMA OPERATIVO - UBUNTO

Taller de Gestión de Memoria en Ubuntu

Introducción

La gestión eficiente de la memoria es un aspecto crítico en el funcionamiento de cualquier sistema operativo. En Ubuntu, basado en Linux, el kernel implementa sofisticados mecanismos de gestión de memoria que permiten optimizar el rendimiento del sistema y el uso de recursos.

Este taller está diseñado para comprender los conceptos fundamentales de la gestión de memoria en Ubuntu y adquirir habilidades prácticas para monitorizar, analizar y optimizar el uso de memoria en sistemas Ubuntu.

Objetivos del Taller

- Comprender los conceptos básicos de gestión de memoria en Ubuntu
- Aprender a usar herramientas para monitorizar el uso de memoria
- Identificar problemas comunes relacionados con la memoria
- Implementar técnicas para optimizar el uso de memoria
- Realizar análisis avanzados de memoria

1. Conceptos Básicos de Memoria en Ubuntu

1.1 Tipos de Memoria

En Ubuntu (y otros sistemas Linux), la memoria se clasifica principalmente en:

- RAM física: La memoria de acceso aleatorio disponible en el hardware.
- Memoria virtual: Combinación de RAM y espacio de swap en disco.
- Memoria de kernel: Memoria utilizada por el kernel de Linux y sus módulos.
- Memoria de usuario: Memoria utilizada por aplicaciones y procesos de usuario.

1.2 Mecanismos de Gestión de Memoria

- Paginación: División de la memoria en bloques (páginas) de tamaño fijo.
- Swap: Extensión de la memoria virtual usando espacio en disco.
- Overcommit: Asignación de más memoria virtual de la disponible físicamente.

- OOM Killer: Mecanismo que termina procesos cuando el sistema se queda sin memoria.

2. Herramientas para Monitorizar la Memoria

Ejercicio 1: Uso básico de free

La herramienta free muestra información sobre la memoria RAM y swap.
bash
 # Mostrar información de memoria en formato legible
free -h
Explicación:

- La salida muestra varias filas incluyendo total, used, free, shared, buff/cache y available.
- -h presenta los datos en un formato "humano legible" (KB, MB, GB).
- available es la cantidad de memoria que puede ser asignada a procesos sin causar swapping.

Tarea: Ejecuta free en diferentes momentos mientras abres varias aplicaciones y observa cómo cambian los valores.

Ejercicio 2: Monitorización con top y htop

bash
 # Instalar htop si no está disponible
sudo apt-get install htop

 # Ejecutar top
top

 # Ejecutar htop (interfaz mejorada)
htop

Explicación:

- En top, presiona Shift+M para ordenar procesos por uso de memoria.
- En htop, puedes usar F6 y seleccionar MEM% para ordenar por uso de memoria.
- Observa las columnas RES (memoria residente), SHR (memoria compartida) y VIRT (memoria virtual).

Tarea: Identifica los 5 procesos que más memoria consumen en tu sistema.

Ejercicio 3: Análisis detallado con vmstat

bash
 # Mostrar estadísticas de memoria cada 2 segundos, 5 veces
vmstat 2 5

 # Mostrar información detallada sobre swap

vmstat -S M

Explicación:

- vmstat muestra información sobre procesos, memoria, paginación, bloques de E/S, traps y actividad de CPU.
- Las columnas swpd, free, buff y cache indican el uso de memoria.
- Las columnas si y so muestran actividad de swap (in/out).

Tarea: Monitoriza la actividad de swap mientras ejecutas una aplicación que consume mucha memoria, como un navegador con muchas pestañas.

3. Gestión de Swap

Ejercicio 4: Configuración y análisis de swap

bash
 # Ver particiones y archivos de swap
sudo swapon --show

 # Ver estadísticas de uso de swap
cat /proc/swaps

 # Crear un archivo de swap de 1GB
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile

 # Para hacerlo permanente, añadir a /etc/fstab:
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab

Explicación:

- El swap es espacio en disco usado como extensión de la memoria RAM.
- Se puede configurar en una partición dedicada o en un archivo.
- La swappiness (0-100) controla la tendencia del kernel a usar swap.

Tarea: Verifica tu configuración actual de swappiness y modifícala temporalmente:

bash
 # Ver swappiness actual
cat /proc/sys/vm/swappiness

 # Cambiar swappiness temporalmente a 10 (favorece mantener datos en RAM)
sudo sysctl vm.swappiness=10

 # Para hacerlo permanente:
echo 'vm.swappiness=10' | sudo tee -a /etc/sysctl.conf

4. Análisis Avanzado de Memoria

Ejercicio 5: Análisis de memoria por proceso con pmap

bash
 # Encuentra el PID de un proceso (ejemplo: Firefox)
pidof firefox

 # Analiza la memoria utilizada por ese proceso
pmap -x [PID]

Explicación:

- pmap muestra el mapa de memoria de un proceso.
- Permite ver cómo se distribuye la memoria: código, datos, heap, stack, librerías compartidas.
- Las columnas muestran el tamaño total, tamaño residente y memoria privada.

Tarea: Analiza la memoria de un navegador web y de un editor de texto. Compara cómo utilizan la memoria.

Ejercicio 6: Análisis con /proc

bash
 # Ver estadísticas generales de memoria
cat /proc/meminfo

 # Ver información de memoria virtual para un proceso específico
cat /proc/[PID]/status | grep -i vm

Explicación:

- /proc/meminfo contiene información detallada sobre la memoria del sistema.
- Cada proceso tiene su directorio en /proc/[PID]/ con información específica.
- VmRSS muestra la memoria física realmente usada por el proceso.

Tarea: Explora los diferentes campos en /proc/meminfo y busca información sobre:

- MemAvailable
- Cached
- Dirty
- CommitLimit y Committed_AS

5. Problemas Comunes y Soluciones

Ejercicio 7: Simular y gestionar agotamiento de memoria
bash
 # Instalar herramienta stress-ng para simular carga
sudo apt-get install stress-ng

 # Simular carga de memoria (consumir 512MB)
stress-ng --vm 1 --vm-bytes 512M --timeout 30s

Explicación:

- Este comando crea un proceso que intenta consumir 512MB de RAM durante 30 segundos.
- Mientras se ejecuta, monitoriza el sistema con free y top en otra terminal.

Tarea: Observa cómo reacciona el sistema cuando hay poca memoria disponible. Intenta modificar los parámetros para encontrar el punto donde el OOM killer entra en acción.

Ejercicio 8: Limitación de memoria con cgroups

bash
 # Instalar herramientas de cgroups
sudo apt-get install cgroup-tools

 # Crear un grupo para limitar memoria
sudo cgcreate -g memory:limitedmem

 # Establecer límite de 100MB
sudo cgset -r memory.limit_in_bytes=100M limitedmem

 # Ejecutar un proceso dentro del grupo limitado
sudo cgexec -g memory:limitedmem stress-ng --vm 1 --vm-bytes 200M --timeout 30s
Explicación:

- Los cgroups permiten limitar recursos para grupos de procesos.
- En este caso, estamos limitando la memoria disponible para un proceso de stress-ng.
- El proceso intentará usar 200MB pero está limitado a 100MB.

Tarea: Observa qué ocurre cuando el proceso intenta exceder su límite de memoria. ¿Se ejecuta el OOM killer o se comporta de otra manera?

6. Optimización de Memoria

Ejercicio 9: Identificación de fugas de memoria

bash
 # Instalar herramientas de depuración
sudo apt-get install valgrind

 # Crear un programa simple con fuga de memoria (memory.c)
cat > memory.c << 'EOF'
#include <stdlib.h>
#include <stdio.h>

```
int main() {
  int i;
  for(i=0; i<1000; i++) {
    // Asignar memoria sin liberarla (fuga)
    int *p = malloc(1000);
    printf("Asignación %d\n", i);
  }
  printf("Finalizado\n");
  return 0;
}
EOF

# Compilar
gcc memory.c -o memory

# Analizar con valgrind
valgrind --leak-check=full ./memory
```

Explicación:

- Este programa intencionadamente asigna memoria sin liberarla.
- Valgrind es una herramienta que detecta fugas y problemas de memoria.
- El reporte al final mostrará las fugas detectadas.

Tarea: Modifica el programa para corregir la fuga de memoria añadiendo free(p) después de la asignación y comprueba si Valgrind detecta mejoras.

Ejercicio 10: Ajuste de parámetros del kernel

```
bash
# Ver configuración actual de memoria
sudo sysctl -a | grep vm

# Configurar temporalmente algunos parámetros
sudo sysctl vm.dirty_ratio=10
sudo sysctl vm.dirty_background_ratio=5
```

Explicación:

- dirty_ratio: Porcentaje de memoria total que puede contener páginas "sucias" (pendientes de escribir a disco) antes de forzar escritura síncrona.
- dirty_background_ratio: Porcentaje que activa el proceso de escritura en segundo plano.

Tarea: Experimenta con diferentes valores y observa cómo afectan al rendimiento del sistema, especialmente cuando hay operaciones intensivas de E/S.

7. Proyectos Prácticos

Proyecto 1: Sistema de monitorización de memoria
Crea un script que monitorice el uso de memoria y envíe alertas cuando se supere cierto umbral:
bash

```
#!/bin/bash

# Script para monitorizar memoria y enviar alertas
THRESHOLD=80  # Porcentaje
LOG_FILE="/tmp/memory_monitor.log"

while true; do
  MEMORY_USED=$(free | grep Mem | awk '{print $3/$2 * 100.0}')
  DATE=$(date '+%Y-%m-%d %H:%M:%S')

  if (( $(echo "$MEMORY_USED > $THRESHOLD" | bc -l) )); then
    echo "$DATE: ¡ALERTA! Uso de memoria: $MEMORY_USED%" >> $LOG_FILE

    # Top 5 procesos por consumo de memoria
    echo "Procesos consumiendo más memoria:" >> $LOG_FILE
    ps aux --sort=-%mem | head -n 6 >> $LOG_FILE
    echo "----------------------------------------" >> $LOG_FILE
  fi

  sleep 60  # Verificar cada minuto
done
```

Explicación:

- Este script verifica el uso de memoria cada minuto.
- Si el uso supera el 80%, registra una alerta y los procesos que más memoria consumen.
- El script utiliza free, awk y ps para obtener y procesar la información.

Tarea: Mejora el script para que envíe notificaciones al escritorio o correos electrónicos cuando se supere el umbral.

Proyecto 2: Análisis de tendencias de memoria
Crea un script que recopile datos de uso de memoria a lo largo del tiempo y genere informes:

bash

```
#!/bin/bash

# Script para recopilar datos de memoria cada hora
DATA_FILE="/tmp/memory_data.csv"

# Crear archivo con encabezados si no existe
```

```
if [ ! -f $DATA_FILE ]; then
   echo "Timestamp,Total,Used,Free,Cache,Available,SwapTotal,SwapUsed" > $DATA_FILE
fi

while true; do
   # Obtener datos de memoria
   TOTAL=$(free -m | grep Mem | awk '{print $2}')
   USED=$(free -m | grep Mem | awk '{print $3}')
   FREE=$(free -m | grep Mem | awk '{print $4}')
   CACHE=$(free -m | grep Mem | awk '{print $6}')
   AVAIL=$(free -m | grep Mem | awk '{print $7}')
   SWAP_TOTAL=$(free -m | grep Swap | awk '{print $2}')
   SWAP_USED=$(free -m | grep Swap | awk '{print $3}')

   # Registrar datos
   echo "$(date '+%Y-%m-%d
%H:%M:%S'),$TOTAL,$USED,$FREE,$CACHE,$AVAIL,$SWAP_TOTAL,$SWAP_USED" >>
$DATA_FILE

   # Esperar una hora
   sleep 3600
done
```

Explicación:

- Este script guarda datos de memoria en un archivo CSV cada hora.
- Los datos incluyen memoria total, usada, libre, caché, disponible y uso de swap.
- Con estos datos se pueden generar gráficos y análisis de tendencias.

Tarea: Usa herramientas como gnuplot o escribe un script Python para visualizar los datos recopilados y detectar patrones de uso de memoria.

Conclusión

La gestión efectiva de la memoria es crucial para mantener un sistema Ubuntu funcionando de manera óptima. A través de este taller, has aprendido a:

1. Comprender los conceptos básicos de gestión de memoria en Ubuntu
2. Utilizar herramientas para monitorizar y analizar el uso de memoria
3. Identificar y resolver problemas comunes relacionados con la memoria
4. Implementar técnicas de optimización de memoria
5. Desarrollar soluciones prácticas para la gestión de memoria

Con estas habilidades, estarás mejor preparado para diagnosticar problemas de rendimiento relacionados con la memoria y optimizar sistemas Ubuntu para diferentes cargas de trabajo.
Recursos Adicionales.

- Documentación oficial de Ubuntu sobre gestión de memoria
- Man pages: man free, man top, man vmstat, man swapon
- El kernel de Linux y sus parámetros: /proc/sys/vm/
- Herramientas avanzadas: perf, systemtap, bcc/BPF