SISTEMAS OPERATIVOS
Trabajo Práctico Nº 3
Administración de memoria en Windows y Linux

Tema

Administración de Memoria Física, Memoria Virtual, Paginación, Fallos de páginas y DMA (Acceso Directo a Memoria) en Windows y Linux.

Resultados de aprendizaje

Comprender la gestión de la memoria de los sistemas operativos Microsoft Windows y Linux con el fin de monitorearla y así detectar problemas.

Enunciado

La empresa ***Cadena de Bloques*** para la gestión de Criptomonedas necesita información sobre sus servidores en Linux y las PC individuales que tienen Windows para elaborar su estrategia de Seguridad. Elabore un informe que conteste las siguientes consignas, adjunte capturas de pantallas cuando considere necesario. Se recomienda leer los anexos antes de empezar a contestar.

Parte I: Utilizando un equipo con Windows, conteste:

1. Identifique la cantidad total de memoria RAM física, cantidad de memoria disponible y cuánto es utilizado como caché del sistema
2. ¿A qué se refiere la Memoria del Núcleo, Total, Paginada y No Paginada?
3. Verifique la información sobre memoria consumida por un proceso en particular ¿Qué proceso eligió? ¿Cuánta memoria estaba ocupando? (adjunte captura de pantalla).
4. Abra el editor de imágenes de Windows (mspaint.exe). Registre los valores de la memoria usada por este proceso. Luego haga una copia de la pantalla con “Print Screen” y pegue la imagen en el editor, tome notas nuevamente de los valores del proceso y el uso de la memoria. Por último aumente la imagen 500% en Vertical y 500% en horizontal. ¿Qué ocurrió con los valores de la memoria usada?
5. Monitoree la memoria e informe si apareció algún fallo de página.
6. Monitoree la memoria desde el Explorador de Procesos. Por favor adjunte la pantalla y defina el significado de cada campo.
7. ¿Existe el archivo llamado “pagefile.sys” en su equipo, qué indica su tamaño y cite la ubicación?
8. ¿Qué beneficio reportaría ubicar al pagefile.sys en una partición exclusiva? Justifique.
9. ¿Por qué evitaría una memoria virtual 6 veces más grande que la RAM que tenemos? Justifique.

Parte II: Utilizando un equipo con Linux (puede ser una máquina virtual), conteste:

1. Memoria física (expresar los resultados en MiB).
    a. Cantidad total.
    b. Cantidad usada.
    c. Cantidad libre.
    d. Cantidad en buff/cache.
    e. Cantidad disponible.
2. Memoria intercambiada a disco (expresar los resultados en MiB).
    a. Cantidad total.
    b. Cantidad usada.
    c. Cantidad libre.
3. Averiguar con qué comando se puede conocer el tamaño de página. ¿Qué tamaño tiene cada página de su sistema operativo Linux?
4. ¿En qué archivo del subdirectorio proc se encuentra la información sobre la memoria? Explique 3 de sus campos.
5. Si su máquina tiene 800 MiB de memoria disponible y 300 MiB de memoria libre, de repente necesita ejecutar un programa que ya se sabe que en promedio ocupa 600 MiB al iniciar. ¿El sistema operativo empezará a *swappear*?
6. Para comprobar lo anterior, en el anexo 4 se deja un programa cuyo único objetivo es solicitar memoria y a esa memoria la establece en cero. El programa cada 1 segundo va solicitando pedazos de memoria. Entonces, llegará un punto que este proceso haya solicitado más memoria que la disponible y el sistema operativo empezará a intercambiar páginas al disco. Utilice las herramientas vmstat y top para monitorear esta situación. Se recomienda utilizar 2 terminales para que en una se vaya ejecutando el proceso consumidor de memoria y en la otra el programa vmstat o top. En una máquina virtual sin entorno gráfico puede utilizar (alt + f1 y alt + f2).
    a. Utilizando top, responda:
        i. ¿Qué sucede con la columna VIRT y RES de este proceso? ¿Qué representa cada una?
        ii. ¿A cuánto llegó la columna %MEM?
        iii. ¿La columna %CPU se vio muy afectada?
        iv. ¿Qué significan las columnas PR y NI?
    b. Utilizando vmstat, responda:
        i. ¿Qué sucede con la columna si? ¿Qué representa esta columna?
        ii. ¿Qué significan las columnas BI y BO?

Presentación del trabajo

La presentación correcta en tiempo y forma es imprescindible para aprobar el trabajo ya que se considera un resultado de aprendizaje a lograr en la formación profesional del alumno.

El trabajo deberá ser presentado en formato PDF y se deberá subir a través del aula virtual de Sistemas Operativos, con el siguiente formato:

Nota: el documento subido deberá estar rotulado como se especificará a continuación, y contener tanto la primera y como la segunda parte del trabajo.

a) ***Nombre del archivo*:** el nombre del archivo debe tener el siguiente formato:
SOP_TPxx_curso_apellido_integrantes.doc (orden alfabético)
Ejemplo: SOP_TP1_2K.._Perez_Lopez_Martinez.doc

Nota: Sólo uno de los integrantes del grupo deberá subir el trabajo práctico.

b) Carátula: explicitando nombre de la universidad, nombre de la cátedra, Curso, Profesor solicitante, Título del tema a desarrollar, Número del grupo, Nombres y legajos de los integrantes y Fecha de entrega del trabajo.

c) Índice de contenidos: implica expresar los temas desarrollados respetando el orden en que se los solicita en el trabajo.

d) Enunciado: enunciado completo del trabajo práctico entregado por el profesor.

e) Introducción: en la cual los alumnos dejarán constancia del contenido principal (a modo de síntesis), a tratarse en el desarrollo del trabajo.

f) Desarrollo: desarrollo del trabajo práctico (cuerpo principal): en el cual se dará respuesta en forma clara y precisa, a todos los requerimientos planteados en el enunciado. Además deberá incluirse la impresión de las pantallas a partir de las cuales se obtuvieron las correspondientes respuestas. Dichas capturas de pantallas deben identificarse con los apellidos de los alumnos del grupo de trabajo y el curso.

g) Conclusión: la que deberá contener una reflexión grupal en relación a la experiencia adquirida, al crecimiento intelectual y personal obtenido. Además se mencionarán los beneficios logrados como futuros ingenieros en Sistemas de Información al realizar el presente trabajo práctico.

h) Bibliografía: deberá citar el material bibliográfico, revistas o sitios virtuales especificando claramente título, autor y edición de los libros y dirección de páginas consultadas.

i) Fecha de entrega: Será establecida por el docente de cada curso, según el cronograma.

Anexo 1: Introducción al monitoreo de memoria en Windows

Para conocer sobre memoria física en Windows de un equipo podemos acceder de las siguientes formas (Los siguientes pasos se basan en Windows 10).
1) Si necesitamos saber la cantidad de memoria Ram, ingresar a *Panel de control.* Adentro del Panel de control se pueden visualizar las opciones de configuración en 3 modos diferentes (arriba a la derecha está la opción para cambiar de modo):

a. Por categoría.
b. Por íconos grandes.
c. Por íconos pequeños.

De acuerdo al modo, la ruta para ingresar es:

- Por categoría: *Sistema y seguridad**Sistema*.
- Por íconos grandes o pequeños: *Sistema*.

2) Si necesitamos saber más sobre la gestión de la memoria, se debe ingresar al Administrador de Tareas de Windows, en la solapa Rendimiento. Para ello, existen al menos 4 maneras de hacerlo:

a) Presionar Ctrl + Shift + Esc.
b) Hacer clic con el botón derecho sobre la barra de tareas y seleccionar el administrador de tareas.
c) Presionar Ctrl + Alt + Delete y hacer clic en el botón del administrador de tareas.
d) Ingresar a Ejecutar, y escribir taskmgr.

3) Si necesitamos saber sobre la memoria física usada por cada uno de los procesos, se debe ingresar al Administrador de Tareas de Windows, en la solapa Procesos.

4) Si necesitamos saber sobre los fallos de página o errores de página, se debe ingresar al Administrador de Tareas de Windows, solapa Rendimiento y luego Monitor de recursos.

Anexo 2: Memoria virtual en Windows

Para casos que la memoria principal es escasa, Windows compensa esta carencia con la utilización de “memoria virtual”.

La memoria virtual combina la memoria RAM (física) de la computadora con uso de espacio temporalmente en el disco rígido. Cuando se intensifica el uso de la RAM, se mueve información de la memoria principal a un espacio llamado archivo de paginación (pagefile.sys). Llevar y traer información al archivo de paginación libera RAM que se necesita para otra tarea.

El archivo pagefile.sys se crea en el momento de la instalación de Windows en la unidad raíz normalmente C: donde se encuentra el boot del sistema, sus atributos son de sistema y oculto.

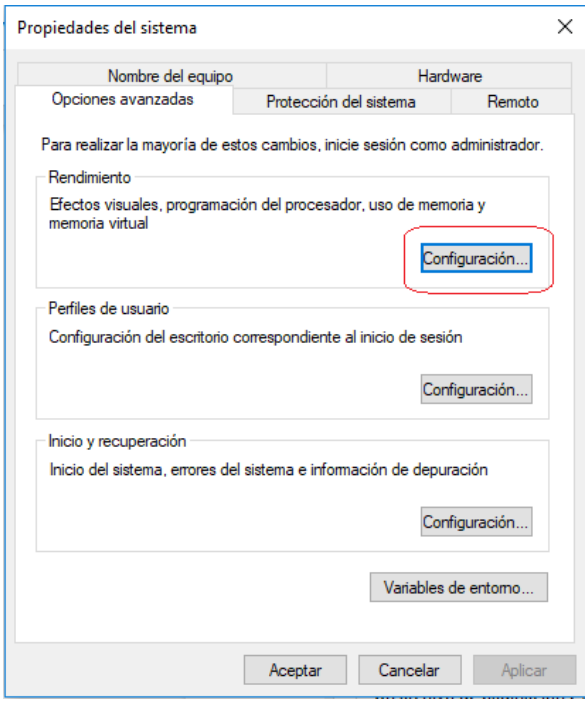
Ante la necesidad de RAM uno puede tentarse a incrementar el tamaño de la memoria virtual, sin embargo, la memoria RAM es mucho más rápida que la lectura de información en el disco por lo que agregar RAM es siempre la mejor solución.

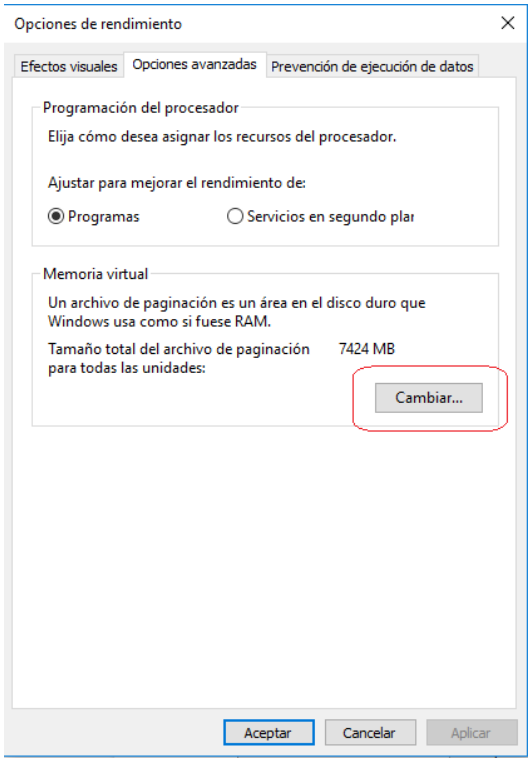
Si recibe mensajes de error advirtiéndole que la memoria virtual es lenta, usted puede optar por agregar RAM o incrementar el tamaño del archivo de paginación. Windows generalmente administra el tamaño de la memoria virtual automáticamente pero usted puede hacerlo manualmente si el valor por defecto no es suficiente. Para hacerlo incremente el tamaño mínimo del archivo de paginación (por defecto igual a la memoria RAM del equipo una vez y medio) y el tamaño máximo (por defecto 3 veces el tamaño de la memoria RAM).

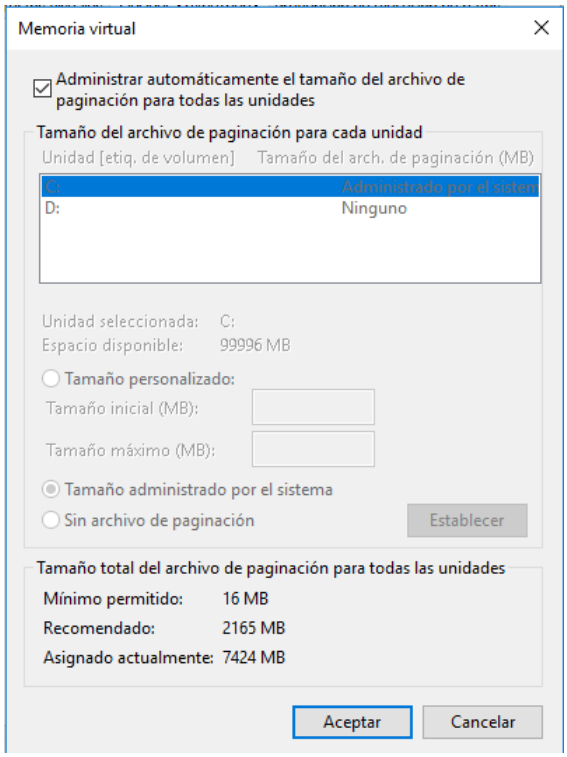
Dados 2 G de RAM:

Mínimo 2 G x 1.5 = 3072 Mb
Máximo 2 G x 3 = 6144 Mb

En Windows 10, para administrar la memoria virtual manualmente se debe acceder a propiedades del sistema:
1) “Inicio” => “Panel de control” => “Sistema y seguridad” => “Sistema” => “Configuración avanzadas del sistema” => “Configuración” => “Opciones avanzadas” => “Cambiar”.







Ahora podemos cambiar la partición que contiene a nuestro archivo de paginación tanto como el tamaño máximo y mínimo, ubicar a nuestro archivo de paginación en una partición creada a tal fin, reporta una mejor eficiencia dado a que evita que el archivo se fragmente.

Aumentar el tamaño de la memoria virtual generalmente no requiere reiniciar el equipo, pero disminuirlo exige el reinicio para que los cambios surtan efecto, en general no se recomienda deshabilitar la memoria virtual como así tampoco eliminar el archivo de paginación.

Anexo 3: Introducción al monitoreo de memoria en Linux

Todos los procesos consumen memoria, algunos más y otros menos. Si un sistema operativo funciona lento, una de las causas más comunes de ello es que la memoria física (o memoria RAM) disponible se esté agotando. Es decir, los procesos están consumiendo más memoria de la disponible realmente. Esto conlleva a que el sistema empiece a utilizar el disco duro para almacenar aquellos datos (o *páginas*) que no entren en la memoria física. Este proceso se denomina *swapping* (o para ser más precisos paginación por demanda). El problema de esta técnica es que el acceso al disco duro es muy lento respecto al acceso a la memoria física.

Entonces es importante poder medir lo que está ocurriendo con la memoria física y con la memoria swap. Empezaremos con la memoria física, a la cual, la podemos dividir en

dos:

OCUPADO	LIBRE

Sin embargo, cuando uno utiliza las herramientas de monitoreo de memoria en Linux, como ser: top, free y vmstat, ellas brindan un poco más de información que a veces suele ser confuso entenderlas. Para comprenderlas, vamos a dividir a la memoria física en lo siguiente:

OCUPADO	BUFF/CACHE	LIBRE

Está claro que la memoria ocupada y libre se explican por sí solas. La memoria ocupada con buffers y la memoria caché, generalmente resumida en memoria *buff/cache* son aquellos datos (o *páginas*) que pueden estar en la swap o que podrían estar próximamente. Si bien este pedazo de memoria contiene datos correctos por lo que se podría considerar como memoria ocupada, realmente se la considera libre pues puede asignarse a otro proceso en caso que se necesite.

A continuación, se presenta una captura parcial de la salida del programa top:

```
top - 16:05:21 up  4:28,  1 user,  load average: 0,73, 0,49, 0,38
Tareas: 287 total,   1 ejecutar,  286 hibernar,    0 detener,    0 zombie
%Cpu(s):  0,6 usuario,  0,7 sist,  0,0 adecuado, 98,6 inact,  0,2 en espera,  0,0 hardw int,  0,0 s
KiB Mem :  8101348 total,   763132 free,  4637492 used,  2700724 buff/cache
KiB Swap:  8312828 total,  8310276 free,     2552 used.  2474288 avail Mem
```

Si analizamos la cuarta línea:

```
KiB Mem: 8101348 total, 763132 free, 4637492 used, 2700724 buff/cache.
```

Podemos encontrar los 4 valores de memoria analizados anteriormente: *total*, *libre*, *ocupada* y *buff/cache*. Luego, en la quinta línea vienen los valores de la memoria swap, es decir, aquella porción de la memoria que está en el disco duro:

```
KiB Swap: 8312828 total, 8310276 free, 2552 user. 2474288 avail Mem
```

Podemos observar el total, lo ocupado y lo libre de la memoria swap. Al final de esta línea se encuentra la cantidad de memoria disponible: *avail Mem*. Este valor es una estimación de cuánta memoria disponible hay para que empiece un nuevo proceso sin *swapping*. A continuación un gráfico para entender el concepto:

OCUPADO	BUFF/CACHE	LIBRE

MEMORIA DISPONIBLE (ESTIMADA)

Uno podría pensar que la memoria disponible siempre debería ser la suma entre la memoria libre más la memoria buff/cache, sin embargo, esto no es así, de hecho el kernel utiliza un algoritmo bastante complejo que tiene en cuenta varios factores para lograr una mejor estimación.

Anexo 4: Programa que solicita memoria en Linux

A continuación se presenta el código fuente de un programa desarrollado en el lenguaje c que solicita memoria cada un segundo hasta un máximo dado. Recibe 2 parámetros obligatorios: MÁXIMO y SALTO. El primero le indica hasta cuánta memoria solicitar y el segundo cuánto solicitar en cada segundo. Ambos valores deben estar expresados en MiB. Por ejemplo:

```
$ consumidor 1500 20
```

Lo anterior le indicará al programa que se ejecute hasta que consuma 1500 MiB en pasos de 20 MiB cada segundo.

Si se desea ejecutar el programa, se debe copiar el código fuente y guardarlo en un archivo como consumidor.c. Luego, compilarlo con lo siguiente:

```
$ gcc consumidor.c -o consumidor.out
```

Finalmente, ejecutarlo con:

```
$./consumidor.out 1500 20
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char** argv) {
    int max = -1;
    int mb = 0;
    char* buffer;
    int salto = 20;

    if(argc == 3) {
        max = atoi(argv[1]);
        salto = atoi(argv[2]);
    } else {
        printf ("Cantidad de parámetros incorrectos\n");
```

```
        return 1;
    }
    if (max < -1) {
        printf ("El parámetro MAX no puede ser menor que -1.\n");
        return 1;
    }
    if (salto <= 0) {
        printf ("El parámetro SALTO no puede ser menor que 0.\n");
        return 1;
    }

    buffer=malloc(1024 * 1024 * salto);
    while(buffer != NULL && (max == -1 || mb + salto <= max)) {
        memset(buffer, 0, 1024 * 1024 * salto);
        mb += salto;
        printf("Se va consumiendo %d MB\n", mb);
        sleep(1);
        buffer=malloc(1024 * 1024 * salto);
    }
    return 0;
}
```