Universitat de les Illes Balears

GRADO DE INGENIERÍA INFORMÁTICA

ACSII

Practica Monitorización

Andrés Borrás Santos, Xiaozhe Cheng

9 de marzo, 2025

Contents

I Introducción

Para la primera práctica de la asignatura se pide que en un entorno Linux, se practique el uso de monitores para obtener información del sistema. Se ha realizado esta práctica en dos dispositivos con sistemas operativos Ubuntu 24.04.2 LTS, uno instalado como Host SO y el otro virtualizado utilizando la versión VMWork Station 17. Se ha utilizado el lenguaje de Shell Bash para la creación de scripts para automatizar la extracción de datos y posteriormente se han analizado los CSV generados con el lenguaje de programación Python (Libreria pandas), en el cual se han generado las gráficas y procesado dichos datos.

Para todos los análisis estadísticos realizados, se han eliminado las primeras 10 muestras para evitar una alteración de los resultados por intervención humana y/o una caché incompleta por parte del sistema.

Característica los dos sistemas monitoreados:

- Nativo - CPU: i5 10310u, RAM: 16 gb, SO: Ubuntu 24.04 LTS
- Virtualizado - CPU: I5 1340p, RAM: 8 gb, SO: Ubuntu 24.04 LTS, VM: WMWork Station 17, núcleos repartidos: 8 núcleos, 8 hilos

II Monitorización de la CPU

En la primera parte del ejercicio, se pide monitorizar la CPU utilizando el monitor TOP incluido en Ubuntu durante 90 minutos y guardar en un CSV los datos Timestamp y los tiempos de la CPU global, usuario y sistema.

II-A ¿Cuántas CPUs tiene el sistema que se ha monitorizado? ¿De dónde se ha obtenido esa información?

Para la obtención del número de CPU del sistema, se ha utilizado el siguiente comando en la terminal de bash:

```
# nproc devuelve el numero de CPUs
# del sistema
nproc
```

Siendo los valores devueltos: 8 hilos para ambos sistemas Ubuntu.

II-B ¿Cuál es la utilización media de la CPU en modo usuario, sistema y en global?

Tras la recopilación de datos, y mediante un script de Python, se han obtenido los siguientes resultados.

(%)	Usuario	Sistema	Global
Media	11.43	3.91	16.19
Mínimo	1.1	1.2	3.6
Máximo	83.0	33.3	90.9
Desviación Estándard	9.35	1.94	10.35

TABLE I: CPU Stats Host SO

(%)	Usuario	Sistema	Global
Media	0.5436	0.9424	1.4853
Mínimo	0.0	0.0	0.0
Máximo	3.7	7.5	11.9
Desviación Estándard	0.3434	0.4038	0.5436

TABLE II: CPU Stats Virtualized SO

II-C ¿Cómo se comportan las medidas anteriores a lo largo del tiempo de observación? Muestra las tres métricas de forma gráfica

A continuación se mostrarán las gráficas obtenidas del procesamiento de las dos recopilaciones de datos. También, se ha suavizado ligeramente todas las graficas usando el método de Exponential Moving Window, para mejorar la visualización de las gráficas.

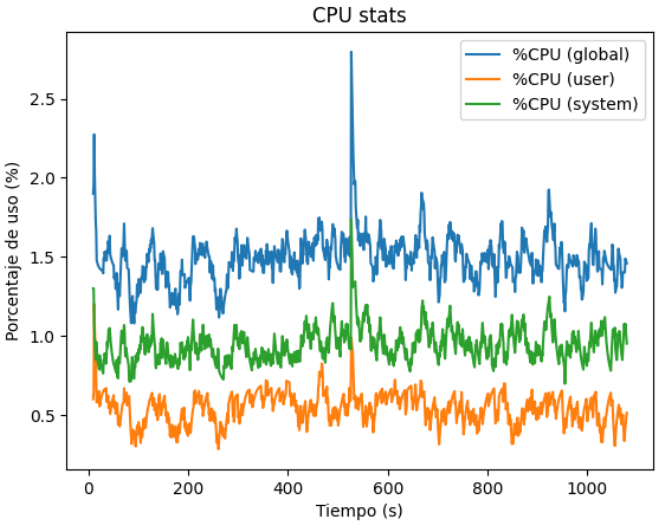

Fig. 1: Virtualized SO CPU Stats

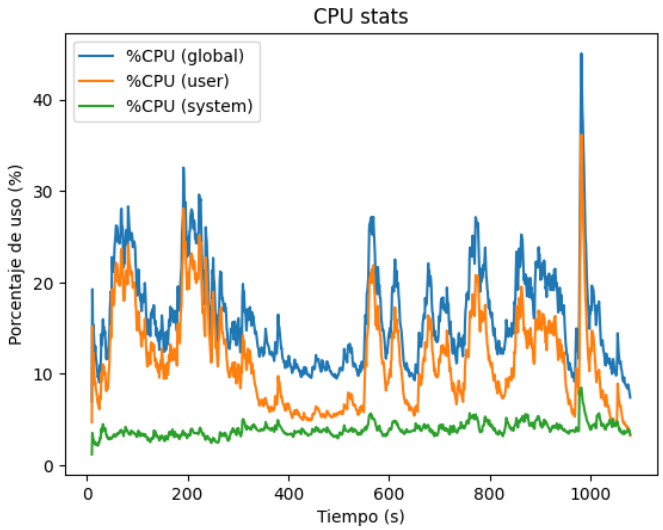

Fig. 2: Host SO CPU Stats

Podemos dar un poco de contexto a las gráficas, la gráfica del sistema virtualizado estuvo única y exclusivamente ejecutando el programa de la práctica, excepto en los picos visibles sobre las 550, 700 y 900 muestras, en donde se revisó el estado del monitoreo y hubo cierta interacción humana.

Por otro lado, en el contexto de la gráfica del sistema instalado en hardware, el usuario comenzó una película. Tras eso, se fue a cenar entre los datos 300 y 600. Tras ello, el usuario se puso a programar, por ello existen picos a partir de la muestra 600, para posiblemente las ejecuciones más pesadas.

Finalmente, podemos observar como el sistema, sin interacción humana, es bastante estable y cuando otros procesos e interrupciones aparecen, este se convierte en errático e impredecible sin un contexto previo. También hay que comentar que el tiempo de sistema es más estable.

II-D ¿Cuál es la sobrecarga provocada por el monitor TOP?

Las sobrecargas se han calculado con un script adicional llamado overhead-monitor, el cual automáticamente nos calcula el overhead de los otros scripts bash y los deja en sus respectivos overhead.txt con el nombre acoplado del archivo bash utilizado.

El cálculo se ha basado en la siguiente fórmula

$$Overhead = \frac{t_{user} + t_{sys}}{t_{real}}$$

que nos ha resultado

(s)	CPU
T. Ejecución	5696.837
T. Monitor	95.828
Overhead	1.6821 %

TABLE III: Overheads Virtual SO

(s)	CPU
T. Ejecución	5660.261
T. Monitor	51.339
Overhead	0.907 %

TABLE IV: Overheads Host SO

III Monitorización de la memoria

El ejercicio nos pide que monitoricemos el PC durante 3 horas con un intervalo de 3 segundos usando `vmstat` y obtener el resultado en un archivo csv.

Para realizar el ejercicio hemos hecho un script llamado `memory-monitor.sh`, que usa un `for` para ir iterando cada muestra y dentro de cada iteración procesamos el output con `awt`.

III-A ¿Qué capacidad total tiene la memoria principal del sistema? ¿De dónde se ha obtenido ese dato?

Con el comando `vmstat` vanilla solo nos da info de swap, free, buff y cache, con los cuales no podemos deducir la capacidad total del sistema. Por lo tanto, hemos usado `vmstat -s` donde la

primera línea está la capacidad total. Y si ejecutamos `vmstat -s | awk 'NR==1' | awk 'print $1'` nos da directamente la cantidad de memoria en el sistema.

(Byte)	Cantidad total de memoria principal
Hardware	16000504
Virtualizado	8082156

III-B ¿Cuál es la utilización media de la memoria? ¿Y la capacidad media utiliza

Igual que en el anterior apartado se ha calculado la utilizacion media de la memoria y la capacidad media utilizada con Python.

(Byte)	Mem. utilizada	% Mem. utilizada
Media	1084207.3359	13.4148
Mínimo	1070680	13.2475
Máximo	1117204	13.8231

TABLE V: CPU Stats Virtualized SO

(Byte)	Mem. utilizada	% Mem. utilizada
Media	2921068.6975	18.25610
Mínimo	1014652	6.3414
Máximo	4094468	25.5896

TABLE VI: CPU Stats Host SO

III-C ¿Cómo se comporta la utilización de la memoria y la capacidad utilizada? Representa estas métricas gráficament

Para calcular la utilización de memoria se nos ha ocurrido dos formas de calcularlos, la primera es

$$m_{used} = m_{total} - m_{free}$$

o la segunda

$$m_{used} = m_{total} - (m_{free} + m_{buff} + m_{cache})$$

Hemos decidido usar la segunda fórmula, porque consideramos que la memoria buff y la memoria cache técnicamente se puede liberar en cualquier momento si el sistema lo pide, por lo tanto, es razonable incluirlos como parte de la memoria libre.

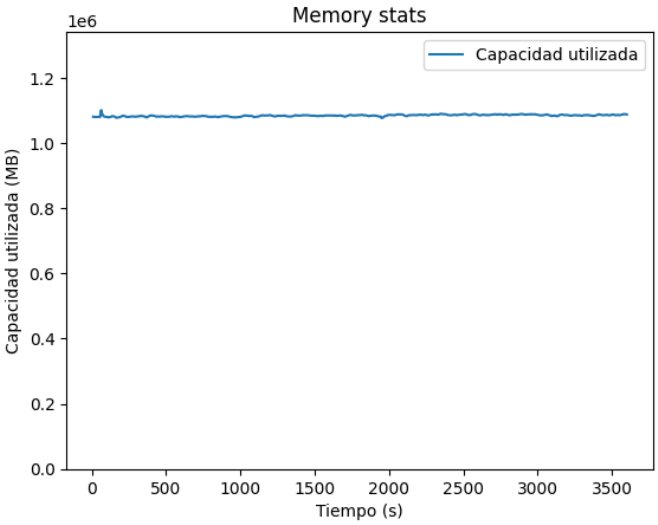

Fig. 3: Virtualized SO Mem Usage Stats

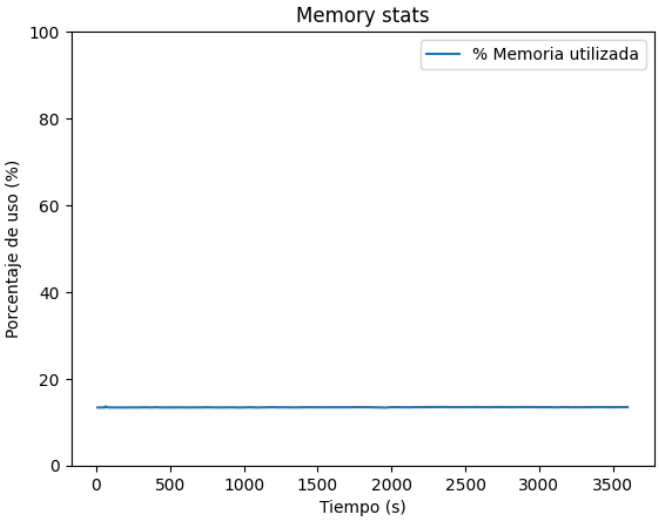

Fig. 4: Virtualized SO Mem % Usage Stats

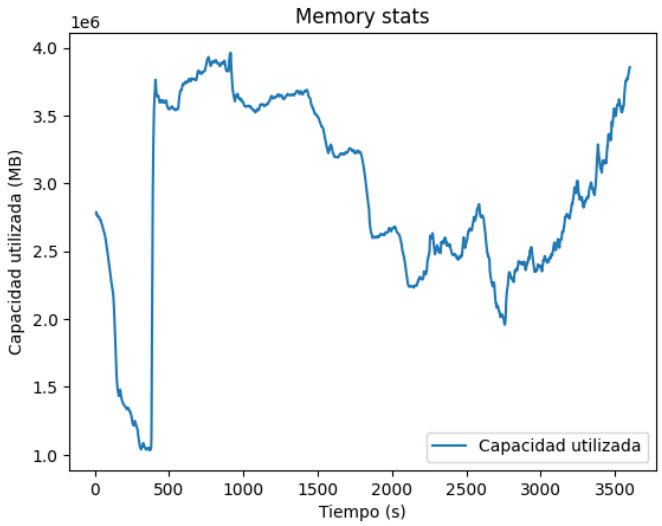

Fig. 5: Host SO Mem Usage Stats

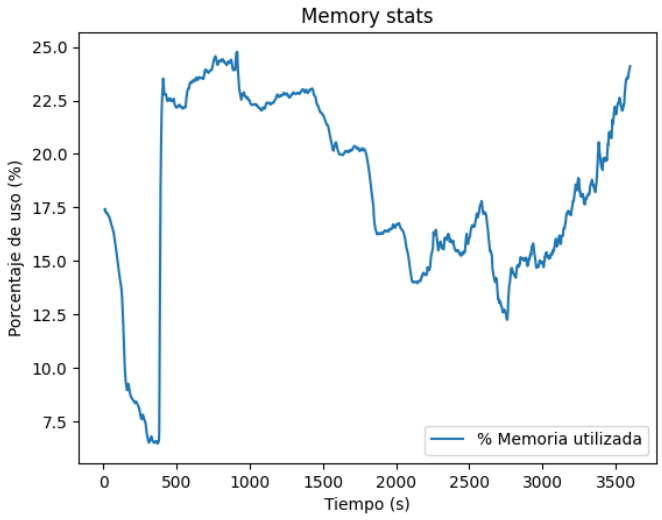

Fig. 6: Host SO Mem % Usage Stats

III-D ¿Cuál es la sobrecarga provocada por el monitor VMSTAT

Como mencionado anteriormente se ha calculado con una script siguiendo la siguiente fórmula:

$$Overhead = \frac{t_{user} + t_{sys}}{t_{real}}$$

que nos ha resultado.

IV Monitorización en paralelo

Para la realización de este ejercicio hemos utilizado el script `paralel-monitor.sh` igual que los anteriores scripts hemos usado un `for` donde en cada iteración llamamos `top` para obtener el '% global CPU' y `vmstat` para obtener 'Capacidad de memoria utilizada' y '% Memoria utilizada' y procesamos los datos con `grep` y `awk` para convertirlos directamente en formato csv.

(s)	Memory
T. Ejecución	10941.577
T. Monitor	162.838
Overhead	1.4882 %

TABLE VII: Overheads Virtual SO

(s)	Memory
T. Ejecución	10895.486
T. Monitor	110.734
Overhead	1.0163 %

TABLE VIII: Overheads Host SO

IV-A Generar un script .sh (o la versión equivalente en bash) para monitorizar la CPU y la memoria principal. Se podrá utilizar el monitor (o los monitores) que se consideren necesarios.

Como mencionado anteriormente hemos hecho el script `paralel-monitor.sh` que está en la carpeta `/actividad-2/scripts`

IV-B Una vez creado el .sh, se monitorizará el sistema durante 2h y se generará el siguiente fichero de salida.

El resultado generado está en la carpeta `/actividad-2/datos_xiao` y `/actividad-2/datos_andres` en formato csv.

IV-C Las muestras se recogerán de forma que la sobrecarga de la monitorización no supere el 10%.

Para monitorizar el overhead de los monitores, hemos creado un script que se basa en pasarle el monitor que queramos calcular el overhead y dentro del script lo ejecutamos con un `time` y aplicamos la fórmula.

$$Overhead = \frac{t_{user} + t_{sys}}{t_{real}}$$

Los resultados que hemos conseguido son los siguientes :

(s)	Paralelo
T. Ejecución	7618.348
T. Monitor	149.244
Overhead	1.95900700 %

TABLE IX: Overheads Host SO

(s)	Paralelo
T. Ejecución	7628.132
T. Monitor	159.406
Overhead	2.08971200 %

TABLE X: Overheads Virtual SO

Como podemos ver, ninguno ha superado el overhead del 10% que pedía el enunciado.

IV-D Gráficas

Como dato extra dejamos unas gráficas sobre las muestras obtenidas en este apartado.

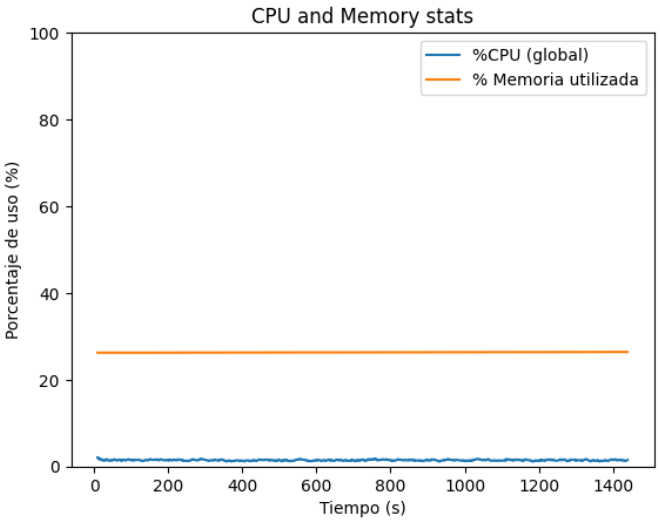

Fig. 7: Virtualized SO

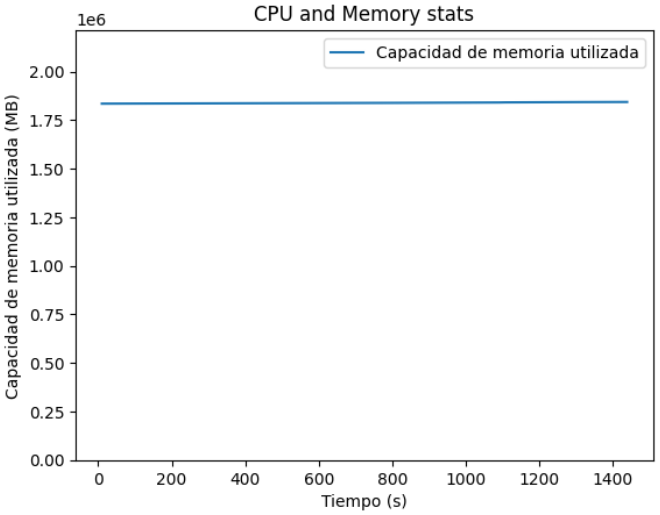

Fig. 8: Virtualized SO

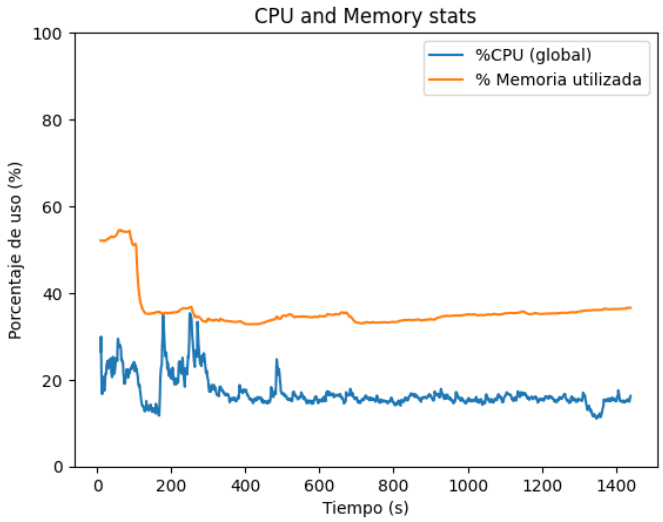

Fig. 9: Host SO

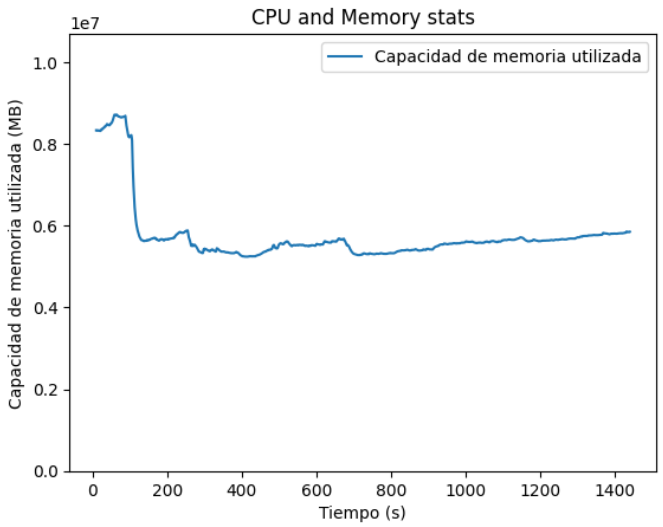

Fig. 10: Host SO

V Sobrecarga basada en detección de eventos

$$Sobrecarga = \frac{a \cdot (\overline{t_e + t_i})}{t_t}$$

- a - la cantidad de accesos para medir el estado de sistema (cantidad de eventos).
- t_e - la cantidad de tiempo que tarda el monitor en recolectar todos los datos necesarios en cada acceso.
- t_i - la cantidad de tiempo que tarda en gestionar la interrupción de cada evento.
- t_t - el tiempo total que ha durado la monitorización.

La idea principal es calcular el tiempo medio que tarda cada evento, es decir, la media del $t_e + t_i$ con la media aritmética.

$$(\overline{t_e + t_i}) = \sum_{i=1}^{n} \frac{t_{e_i} + t_{i_i}}{n}$$

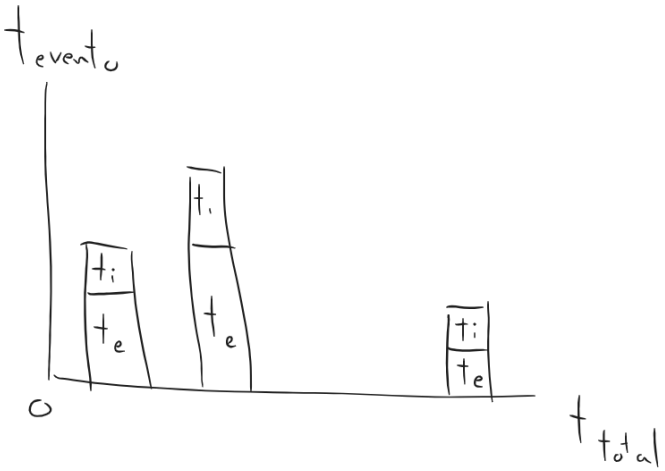

Fig. 11: Dibujo representativo de la idea