Àlex Aguilera

Miguel Fernández

Práctica medidas de rendimiento

**1. Escribir un programa que calcule el tiempo que tarda una llamada a sistema sencilla. Haga una tabla con los tiempos de ejecución que obtenga. ¿Por qué los tiempos de ejecución son tan diferentes?**

El código que hemos usado para completar la tabla siguiente, se encuentra en el fichero syscalls.c. Hemos ejecutado las primeras syscalls N=1000000 veces y la syscall fork() N2= 10000 veces, tal y como dice el enunciado.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Syscall | sbrk(0) | sbrk(inc) | sched\_yield() | getpid() | fork/waitpid |
| Execution time (in microsecs.) | 3092 | 226862 | 302723 | 190193 | 1331454 |

El tiempo de ejecución de cada syscall es diferente y puede variar bastante entre unas y otras. Esto, es debido a que a que algunas llamadas como podrían ser sbrk() o getpid(), son bastante simples y simplemente han de retornar un valor y no hacer ninguna operación de cálculo que pueda incrementar su tiempo de ejecución. Sin embargo, hay otras como ejemplo la llamada a sistema fork(), que debe crear todas las estructuras necesarias para crear un nuevo proceso e inicializarlas debidamente, es por ello que su tiempo de ejecución es considerablemente mayor.

**2. Puede comprobar de alguna manera que los programas ejecutan realmente la llamada a sistema? (Y no aprovechan el resultado devuelto por la llamada anterior)**

Para comprobar que llamadas a sistema utiliza un programas y ver si realmente se aprovechan del resultado devuelto por otra llamada, podemos usar strace (comentado en el forum del Racó). Para ello, simplemente hay que ejecutar en la consola lo siguiente: strace ./syscalls 0/1/2/3/4 e ir viendo si ejecuta esa syscall o no.

En caso de sbrk(), utilitza la syscall brk. Para sched\_yied() y getpid(), utiliza esta misma. En el caso de fork(), utiliza clone() con los parámetros vistos en clase de teoría y para el caso de waitpid(), utilitza wait4(-1,NULL,0,NULL).

**3. En la última transparencia del tema Virtualización-Sincronización-MesuresdeRendiment (transp. 50) le pedimos que haga un programa que escriba en el disco 500MBytes, midiendo el tiempo que tarda en hacerlo. Desde un usuario no privilegiado (no root), ejecute su programa sobre un archivo en el disco de la máquina que se utiliza para este laboratorio.**

Con los códigos individuales de cada estudiante, hemos obtenido los siguientes diagramas:

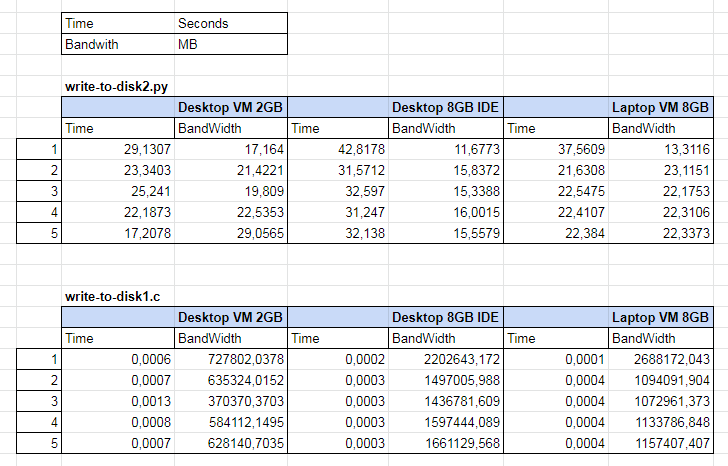


Figura 1. Resultados

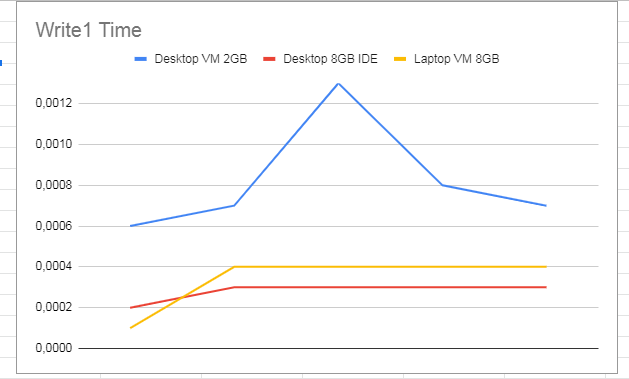


Figura 2. Gráfica Tiempo write-to-disk1.c (segundos)

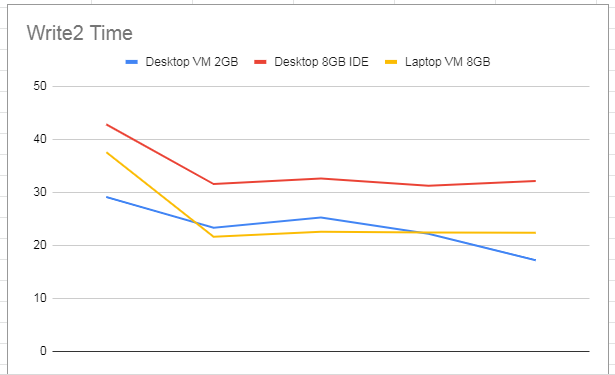


Figura 3. Gráfica Tiempo write-to-disk2.py (segundos)