Introducción a la evaluación del rendimiento (Ubuntu)

Tarea 1:

- **uptime:** enseña una línea con la siguiente información: la hora actual, cuánto tiempo lleva en marcha el sistema, cuántos usuarios están conectados y las cargas promedio del sistema (número de procesos en estado runnable o uninterruptable en media) para los últimos 1, 5 y 15 minutos. Muestra la salida en formato HH:MM:SS up “tiempo” min, “n_usuarios” user, load average: “load1”, “load5”, “load15”.

- **vmstat:** muestra estadísticas de memoria virtual. Concretamente enseña información sobre procesos, memoria, paginación, entrada/salida de bloque, interrupciones de software, discos y CPU. La salida es en formato de tabla, con las dos primeras filas agrupando las variables acerca de las que se enseña información por secciones (procesos, memoria, memoria intercambiada de disco, input/output, sistema y CPU, y a continuación una fila para cada actualización que se hace del comando (se puede usar como argumento cada cuánto tiempo se quiere que se ejecute).

- **ps:** por defecto muestra información sobre los procesos activos relacionados con el usuario actual y asociados con la terminal donde se ha invocado ps y de éstos da el PID (process ID), el TTY (nombre de fichero de la terminal), el tiempo de CPU que han acumulado en formato HH:MM:SS y el nombre del ejecutable (CMD). Con las opciones se pueden mostrar más procesos.

- **iostat:** este comando informa de estadísticas de cpu y de input/output para dispositivos y particiones. Concretamente se usa para monitorear la carga de los dispositivos de entrada/salida del sistema observando el tiempo que están en activo en relación a su promedio de transferencias. La salida también es en formato de tabla: primero tenemos una enseñando información del porcentaje de CPU usado a distintos niveles, y luego otra tabla donde para cada dispositivo o partición usados por el sistema, aparece la cantidad de datos que escriben por segundo, el número de transferencias por segundo que realizan,...

- **powertop:** es un comando que sirve para diagnóstico de consumo de energía y manejo de energía. Contiene un modo interactivo que permite experimentar con configuraciones de manejo de energía. La salida también es en formato de tabla, se

muestra una fila por cada componente que puede estar consumiendo del sistema, con una columna estimando la energía que gasta y el número de wakeups (desbloqueos) que hace por segundo, juntamente con una columna describiendo la categoría a que pertenece y una de descripción del componente.

Tarea 2:

2. Tanto el comando iostat como el comando vmstat proporcionan información relacionada con el rendimiento del sistema, pero se centran en diferentes aspectos del rendimiento.

Iostat se utiliza principalmente para monitorear el rendimiento del disco y los dispositivos de almacenamiento. Por otro lado, vmstat se utiliza para monitorear el uso de la memoria, la CPU y los procesos en ejecución.

Ambos comandos pueden ser útiles para determinar la causa de problemas de rendimiento en el sistema. Si el sistema está experimentando una alta carga de disco, iostat puede ser útil para identificar los procesos que están generando la mayor cantidad de operaciones de entrada y salida. Si el sistema está experimentando una alta carga de CPU o problemas de memoria, vmstat puede ser útil para identificar los procesos que están utilizando la mayor cantidad de recursos.

Un ejemplo de cómo utilizar ambos comandos para monitorear el rendimiento del sistema es por un lado ejecutar el comando iostat -x 5, lo que mostrará las estadísticas de uso del disco cada 5 segundos, incluyendo la tasa de transferencia de datos y los tiempos de respuesta promedio, y por otro lado ejecutar el comando vmstat 5, lo que mostrará las estadísticas de uso de la CPU y la memoria cada 5 segundos, incluyendo la cantidad de procesos activos e inactivos, la cantidad de memoria libre y en uso, y el porcentaje de uso de la CPU.

3. Si entramos en el directorio /proc, identificamos que hay un fichero llamado vmstat, como el propio comando, que si lo leemos, nos puede proporcionar la misma información que el comando: uso de memoria virtual, procesos, memoria, E/S y actividad de CPU. La única diferencia que hay entre ellos es que el archivo es un archivo de texto donde en cada línea hay una sola característica, y en cambio el comando ofrece una tabla formateada además de la posibilidad de configurar la salida e ir sacando muestreos por intervalos.

4. Concretamente he decidido utilizar un programa de multiplicación de matrices con Java y con C:

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int n1, m1, n2, m2, i, j, k;
        int[][] mat1 = new int[10][10];
        int[][] mat2 = new int[10][10];
        int[][] result = new int[10][10];

        // Pedir las dimensiones de las matrices
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce el número de filas de la primera matriz: ");
        n1 = scanner.nextInt();
        System.out.print("Introduce el número de columnas de la primera matriz: ");
        m1 = scanner.nextInt();
        System.out.print("Introduce el número de filas de la segunda matriz: ");
        n2 = scanner.nextInt();
        System.out.print("Introduce el número de columnas de la segunda matriz: ");
        m2 = scanner.nextInt();

        // Verificar si las matrices son compatibles para la multiplicación
        if (m1 != n2) {
            System.out.println("Error: Las matrices no son compatibles para la multiplicación");
            return;
        }

        // Pedir los elementos de la primera matriz
        System.out.println("Introduce los elementos de la primera matriz:");
        for (i = 0; i < n1; i++) {
            for (j = 0; j < m1; j++) {
                mat1[i][j] = scanner.nextInt();
            }
        }

        // Pedir los elementos de la segunda matriz
        System.out.println("Introduce los elementos de la segunda matriz:");
        for (i = 0; i < n2; i++) {
            for (j = 0; j < m2; j++) {
                mat2[i][j] = scanner.nextInt();
            }
        }

        // Calcular el producto de las dos matrices
        for (i = 0; i < n1; i++) {
            for (j = 0; j < m2; j++) {
                result[i][j] = 0;
                for (k = 0; k < m1; k++) {
                    result[i][j] += mat1[i][k] * mat2[k][j];
                }
            }
        }

        // Imprimir el resultado
        System.out.println("El producto de las dos matrices es:");
        for (i = 0; i < n1; i++) {
            for (j = 0; j < m2; j++) {
                System.out.print(result[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Fig. 1: Multiplicación de matrices con Java

```c
#include <stdio.h>

#define MAX_SIZE 10

int main() {
    int n1, m1, n2, m2, i, j, k;
    int mat1[MAX_SIZE][MAX_SIZE], mat2[MAX_SIZE][MAX_SIZE], result[MAX_SIZE][MAX_SIZE];

    // Pedir las dimensiones de las matrices
    printf("Introduce el número de filas de la primera matriz: ");
    scanf("%d", &n1);
    printf("Introduce el número de columnas de la primera matriz: ");
    scanf("%d", &m1);
    printf("Introduce el número de filas de la segunda matriz: ");
    scanf("%d", &n2);
    printf("Introduce el número de columnas de la segunda matriz: ");
    scanf("%d", &m2);

    // Verificar si las matrices son compatibles para la multiplicación
    if (m1 != n2) {
        printf("Error: Las matrices no son compatibles para la multiplicación\n");
        return 0;
    }

    // Pedir los elementos de la primera matriz
    printf("Introduce los elementos de la primera matriz:\n");
    for (i = 0; i < n1; i++) {
        for (j = 0; j < m1; j++) {
            scanf("%d", &mat1[i][j]);
        }
    }

    // Pedir los elementos de la segunda matriz
    printf("Introduce los elementos de la segunda matriz:\n");
    for (i = 0; i < n2; i++) {
        for (j = 0; j < m2; j++) {
            scanf("%d", &mat2[i][j]);
        }
    }

    // Calcular el producto de las dos matrices
    for (i = 0; i < n1; i++) {
        for (j = 0; j < m2; j++) {
            result[i][j] = 0;
            for (k = 0; k < m1; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }

    // Imprimir el resultado
    printf("El producto de las dos matrices es:\n");
    for (i = 0; i < n1; i++) {
        for (j = 0; j < m2; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Fig. 2: Multiplicación de matrices con C

Tras ejecutar vmstat con actualizaciones cada 5 segundos durante la ejecución de cada programa para una multiplicación de dos matrices 3x3 con los mismos números hemos obtenido lo siguiente:

```
xavi@xavi-VirtualBox:~$ vmstat 5
procs -----------memoria---------- ---swap-- -----io---- -sistema-- ------cpu-----
 r  b   swpd  libre búfer caché   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 3720024 104208 2335460    0    0   211    71  295  377  3  1 96  0  0
 0  0      0 3715740 104208 2335520    0    0     0    10 1067  973  1  0 99  0  0
 1  0      0 3715740 104216 2335520    0    0     0   110  670  288  0  0 100  0  0
 0  0      0 3715740 104224 2335520    0    0     0     3  749  391  0  0 100  0  0
 0  0      0 3715488 104224 2335520    0    0     0     2  661  273  0  0 100  0  0
 0  0      0 3715488 104232 2335516    0    0     0     3  707  411  0  0 100  0  0
 0  0      0 3715488 104232 2335516    0    0     0     2  740  479  0  0 100  0  0
 0  0      0 3722804 104240 2335648    0    0    26    13  908  807  1  0 99  0  0
^C
```

Fig. 3: caso Java

```
xavi@xavi-VirtualBox:~$ vmstat 5
procs -----------memoria---------- ---swap-- -----io---- -sistema-- ------cpu-----
 r  b   swpd  libre búfer caché   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 3728204 101980 2263444    0    0   310    96  336  435  4  1 95  0  0
 0  0      0 3727704 101980 2263568    0    0     0     0 1054 1099  1  0 99  0  0
 0  0      0 3727992 101988 2263492    0    0     0    24  764  409  0  0 99  0  0
 0  0      0 3727768 101996 2263536    0    0     0    30  653  304  0  0 100  0  0
 0  0      0 3727768 102004 2263528    0    0     0     8  689  306  0  0 100  0  0
 0  0      0 3720048 102188 2264896    0    0   253     1  835  617  1  0 99  0  0
 0  0      0 3720048 102196 2264896    0    0     0    46  690  424  0  0 100  0  0
 0  0      0 3720048 102204 2264896    0    0     0     8  716  505  0  0 100  0  0
```

Fig. 4: caso C

Como podemos observar, la columna que hay que destacar es la de memoria libre, donde vemos que ha habido más cuando hemos ejecutado el programa en C. Esto es debido a que Java utiliza un modelo de gestión de memoria basado en recolección de basura, que mantiene un seguimiento de cada objeto creado y puede resultar en un mayor uso de memoria durante la ejecución del programa, mientras que C te permite gestionar la memoria dinámicamente y de forma contigua. En cuanto a uso de CPU, solo cabe comentar que está muy desocupada.

5. Ahora realizaremos el mismo experimento pero con una entrada bastante grande para ambos programas a diferencia de anteriormente (he bajado el tiempo de sampling a 2 segundos):

```
xavi@xavi-VirtualBox:~/Escritorio$ vmstat 2
procs -----------memoria---------- ---swap-- -----io---- -sistema-- ------cpu-----
 r  b   swpd  libre búfer caché   si   so    bi    bo   in   cs us sy id wa st
 0  0   1036 1900744 168272 2146948    0    0    33    54  306  466  4  1 96  0  0
 0  0   1036 1895468 168272 2146980    0    0     0     0 2079 2155  4  1 95  0  0
 0  0   1036 1895468 168280 2146980    0    0     0    22 1026  861  0  0 100  0  0
 0  0   1036 1895468 168280 2146980    0    0     0     0  964  720  0  0 100  0  0
 0  0   1036 1886156 168280 2146980    0    0     0    56 1097  910  2  0 98  0  0
 0  0   1036 1913008 168280 2146980    0    0     0     0 2149 13366  3  1 96  0  0
 0  0   1036 1912756 168280 2146992    0    0     0     0 1341 1123  1  0 99  0  0
^C
xavi@xavi-VirtualBox:~/Escritorio$ vmstat 2
procs -----------memoria---------- ---swap-- -----io---- -sistema-- ------cpu-----
 r  b   swpd  libre búfer caché   si   so    bi    bo   in   cs us sy id wa st
 1  0   1036 1911548 168384 2147092    0    0    33    54  305  464  4  1 96  0  0
 0  0   1036 1911308 168384 2147132    0    0     0     0 1332 1581  2  0 98  0  0
 0  0   1036 1911308 168384 2147132    0    0     0     4  903  789  0  0 100  0  0
 0  0   1036 1911308 168384 2147132    0    0     0     0  919 1008  0  0 100  0  0
 1  0   1036 1911308 168392 2147132    0    0     0     8  894  811  0  0 100  0  0
 1  0   1036 1911308 168392 2147188    0    0     0   324  935  770  0  0 100  0  0
```

Fig. 5: caso Java y C respectivamente para entrada de dos matrices 100x100

En este caso, vemos que ambos programas han recurrido a la utilización de memoria virtual, y que C sigue usando menos memoria. Por otra parte, observamos que Java ha hecho más uso de la CPU (columna id, que indica cuánto tiempo ha permanecido libre). Esto puede ser debido al overhead que supone que java se ejecute una máquina virtual, o debido al sistema de gestión de memoria de Java. Por último, cabe destacar que durante la ejecución del programa en Java ha habido una cantidad sobresaliente de cambios de contexto (columna cs), concretamente 13366, esto puede ser debido a que Java usa hilos para gestionar ciertas tareas, como el recolector de basura y el manejador de señales, y cada hilo requiere su propio contexto de ejecución.

6. Podríamos medir el uso de recursos de un programa web con un comando llamado top. Este comando muestra información en tiempo real sobre los procesos en ejecución del sistema y su uso de recursos. Cuando se ejecuta, top muestra una lista de procesos ordenados por el uso de CPU. Para cada proceso, se muestran detalles como el ID del proceso, el usuario que lo ejecuta, el porcentaje de CPU y memoria que está utilizando, y el tiempo que ha estado en ejecución. Por tanto lo suyo sería identificar qué proceso o procesos está/n implicado/s en el uso de este programa web y llevar a cabo una monitorización.

Tarea 3:

[1]https://geekland.eu/usar-entender-monitor-de-recursos-top/