Universitat de les Illes Balears

Escola Politècnica Superior

21719 - Avaluació del Comportament de Sistemes Informàtics.

Práctica 2: Tema 3 - Benchamarking.

Khaoula Ikkene
Grupo 102
khaoula.ikkene1@estudiant.uib.cat

ENUNCIADO DE LA PRÁCTICA

Para la evaluación del sistema actual, se utilizará la carga Sysbench CPU con un porcentaje de uso de la CPU del 50%, la cual se ejecutará en el sistema actual y se harán uso de las técnicas de monitorización ya aprendidas en la práctica anterior. De este modo, se pide responder a las siguientes preguntas:

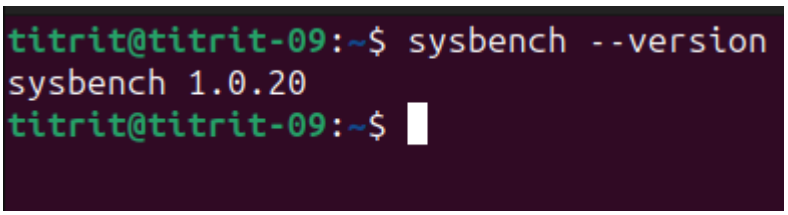
Pregunta 1

Explica con detalle cómo es el diseño y la implementación del experimento para evaluar el sistema actual. Se deben justificar las decisiones tomadas, desde el número de muestras que se van a tomar hasta qué monitores se van a lazar y por qué.

En primer lugar, proporcionaré los siguientes datos y detalles sobre mi portátil que nos ayudarán a tomar decisiones más informadas. Al ejecutar el comando $cat /proc/cpuinfo, obtenemos la siguiente información:

model name: 12th Gen Interl(R) Core(TM) i7-12650H
CPU cores: 6
Threads per core: 1
CPU frecuency: 2688.010 MHz
RAM: 10GiB
Además de que el sistema operativo es Linux, ejecutado sobre una imagen de Ubuntu (20.04.6 LTS) sobre nuestra máquina virtual (Virtualbox 7.0.14).

En cuanto a Sysbench, cuento con la siguiente versión:



Con la ayuda del comando $sysbench man obtenemos la siguiente información sobre el comando que tenemos que ejecutar:
$sysbench cpu --max-prime=<> --threads=<> --time=0 --events=<cantidad_de_carga> run

Donde:
--threads=N: Define el número de hilos a utilizar. Cuando se establece en 0, no hay límite.
--events=N: Establece un límite para el número total de eventos (cálculos de números primos).
--max-prime=N: define un límite para el cálculo de números primos.

Ahora procederemos a definir los valores de estos parámetros.

En cuanto a la cantidad máxima de hilos, dado que solo se utilizará el 50% de la CPU, se establecerá a 3.

El parámetro "time" es independiente del trabajo y los cálculos que realiza el comando Sysbench. Representa el límite de tiempo para la ejecución total de la prueba en segundos. Según el enunciado, se le asigna un valor de 0, lo que significa que es ilimitado; es decir, la prueba de Sysbench durará el tiempo que sea necesario para completar los cálculos de números primos.

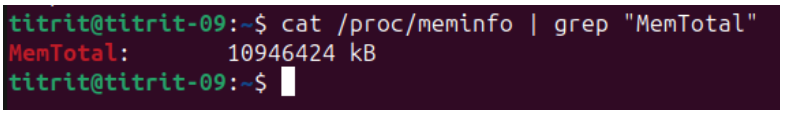
Para el parámetro "*events*", su valor será el mismo que el valor de los números primos. Ya que de esta forma forzamos esta versión de sysbench a trabajar como las versiones antiguas.

Una vez que tenemos los comandos especificados con los parámetros adecuados, podemos discutir qué monitor utilizar. Dado que la principal solicitud es monitorear la CPU, la opción más apropiada sería utilizar el monitor TOP. Sin embargo, también se requiere información relacionada con la memoria, especialmente el porcentaje de uso medio.

Según la práctica previa, sabemos que el cálculo del uso medio de la memoria se realiza de la siguiente manera:
*% Uso Medio de Memoria = (Memoria Total - Memoria Libre) * 100 / Memoria Total.*

Es decir, si disponemos de la cantidad de memoria total, solo nos faltará la información sobre la memoria libre durante la toma de muestras. Y esta información ya nos la proporciona el monitor TOP. Por lo tanto, al utilizar dicho monitor, obtendremos toda la información que necesitamos.

Para obtener la cantidad de memoria total de nuestro sistema, ejecutamos el siguiente comando:



La determinación del número de muestras y su intervalo requiere consideraciones fundamentales como la estabilidad del sistema. Se supone que en condiciones de estabilidad, la necesidad de tomar un número significativo de muestras disminuye, dado que el comportamiento del sistema tiende a mantenerse constante.

Por lo tanto, realizaremos un par de pruebas con Sysbench, y como era de esperar, el tiempo de ejecución de cada una de ellas fue aproximado, es decir, no variaba mucho de una carga a otra.
Así que, podemos optar a tomar 15 muestras con intervalos de 2 segundos para capturar los detalles significativos, aunque con 10 muestras ya sería suficiente.
En total, esto nos permitirá obtener 60 muestras de cargas y, por lo tanto, poder evaluar con precisión el rendimiento de la CPU y la memoria bajo diferentes niveles de carga, proporcionando una visión completa de su funcionamiento.
Para ejecutar los comandos Sysbench de estas cargas y redireccionar su salida a un fichero usaremos el siguiente script.sh:

```
# Definir el arreglo carga
carga=(30000 60000 90000 120000)
# Bucle para ejecutar sysbench cpu con diferentes cargas
    for carga_valor in "${carga[@]}"; do
    for i in {1..15}; do
sysbench cpu --cpu-max-prime="$carga_valor" --threads=3 --time=0 --events="$carga_valor" run >> CARGA.txt
        sleep 2
    done
     echo "Ejecución $carga_valor completa" >> CPU.txt
done
#acabar la monitorizacion
    pkill -f code.sh
```

Para las muestras de la CPU y la memoria, no hemos especificado un número fijo para cada nivel de carga, sino simplemente un límite superior que asegure la monitorización completa de todas las cargas, con intervalos de 2 segundos. Por lo tanto, el número de muestras recolectadas para cada carga será proporcional a la duración de su ejecución. Utilizaremos el siguiente script básico para recolectar estas muestras:

```
#!/bin/bash
#recogemos las muestras de CPU y memoria
for i in {1..4400}; do
    top -b -n 1 | grep -i -E "Cpu\(s\)|MiB Mem" >> CPU.txt
    sleep 2
done
```

Una vez que tenemos nuestras muestras, ejecutaremos el script *CPUMEMusage.py* para tratar los datos y generar los ficheros de salida.
Y para los valores de tiempo de respuesta y productividad usaremos el script convertir_csv.py

CPUMEMusage.py

```
import pandas as pd
def main():
    CPU_id = []
    memTotal = 10946424
    free_mem = []
    with open('CPU.txt', 'r') as fichero:
    for i, linea in enumerate(fichero):
        if i % 2 == 0:  # Si el índice es par, procesar datos de CPU
     CPU_id.append(round(100 - float(linea[35:41].replace(',', '.')), 1))
        else:  # Si el índice es impar, procesar datos de memoria
    free_mem.append((memTotal-(float(linea[28:35].replace(',', '.')))*1048)*100/memTotal)
    #se multiplica por el factor 1048 para pasar de MiB a kB

    # Crear DataFrame para datos de CPU
    cpu_data = {'% CPU (global)': CPU_id}
    df_cpu = pd.DataFrame(cpu_data)

    # Crear DataFrame para datos de memoria
    memory_data = {'%Mem': free_mem}
    df_memory = pd.DataFrame(memory_data)
    # Escribir los datos en archivos CSV
    df_cpu.to_csv('90kcpuprime.csv', sep=';', index=False)
    df_memory.to_csv('90Kmemprime.csv', sep=';', index=False)

if __name__ == '__main__':
    main()
```

convertir_csv.py

```python
import re
import csv

def main():
    # Abrir el archivo de entrada
    with open('CARGA.txt', 'r') as file:
        data = file.read()

    # Encontrar todos los valores de events per second y total time
    events_per_second = re.findall(r'events per second:\s+([\d.]+)', data)
    total_times = re.findall(r'total time:\s+([\d.]+)s', data)

    # Escribir los valores en archivos CSV
    with open('events_per_second.csv', 'w', newline='') as events_csvfile:
        writer = csv.writer(events_csvfile)
        writer.writerow(['Events per Second'])
        for eps in events_per_second:
            writer.writerow([eps])

    with open('tiempoRespuesta.csv', 'w', newline='') as time_csvfile:
        writer = csv.writer(time_csvfile)
        writer.writerow(['Total Time (s)'])
        for time in total_times:
            writer.writerow([time])

if __name__ == '__main__':
    main()
```

Pregunta 2

¿Cómo se comporta el sistema actual si variamos la carga varía en 30000, 60000, 90000 y 120000 números primos? ¿Cómo es el comportamiento del tiempo de respuesta y la productividad? Indica el valor para cada una de las ejecuciones del experimento y el valor medio (de todos los experimentos).

El tiempo de respuesta ya nos lo ofrece el Sysbench, y la productividad se puede calcular utilizando la fórmula: $Productividad = \frac{carga}{tiempo\ total\ de\ prueba}$.
O lo que es equivalente a coger events/s que ya proporciona el sysbench.

```
CPU speed:
    events per second:    28.48

General statistics:
    total time:                          7022.9708s
    total number of events:              200000

Latency (ms):
         min:                                    16.48
         avg:                                    35.11
         max:                                   331.70
         95th percentile:                        44.98
         sum:                                 7022182.19
```

En las tablas siguientes, se presentarán los tiempos de respuesta junto con sus valores promedio, calculados utilizando la media aritmética. Asimismo, se mostrarán los valores de productividad

con sus respectivos promedios, utilizando la media armónica, debido a que el tiempo se encuentra en el denominador

Se puede notar que a medida que incrementamos la carga, se realizan más cálculos complejos de

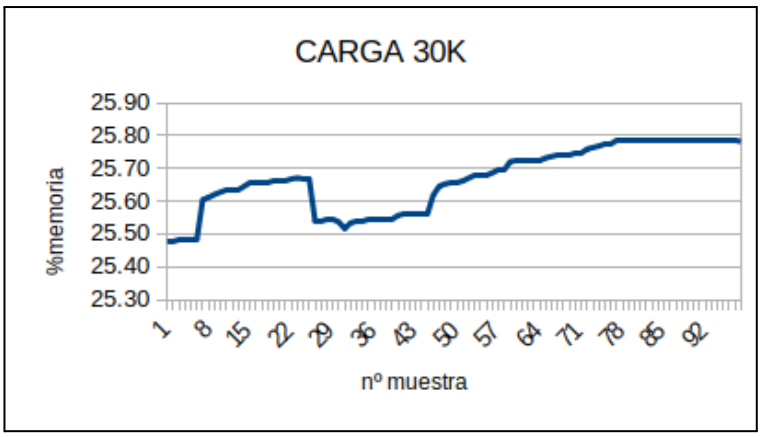
Tiempo de respuesta(s)			
30K	60K	90K	120K
12.5459	63.7764	301.5039s	625.3388
12.4791	63.0039	380.4399	567.357
12.6837	62.9513	378.4253	624.5562
12.6694	63.3203	299.1143	653.8245
13.0908	69.0487	334.6107	651.895
12.7041	68.4112	340.9021	638.0814
12.7833	63.6489	306.1694	483.7049
12.6389	64.1872	337.6024	649.695
12.7189	63.1727	171.6111	581.1678
12.5154	66.5078	173.6003	579.7987
12.7907	63.4715	193.8966	586.8133
12.4071	73.426	323.537	584.284
12.8354	63.349	243.0269	618.2291
12.7758	66.4175	257.9929	398.114
13.3958	73.8057	304.073	340.5836
12.74	**65.9**	**288.93**	**572.23**

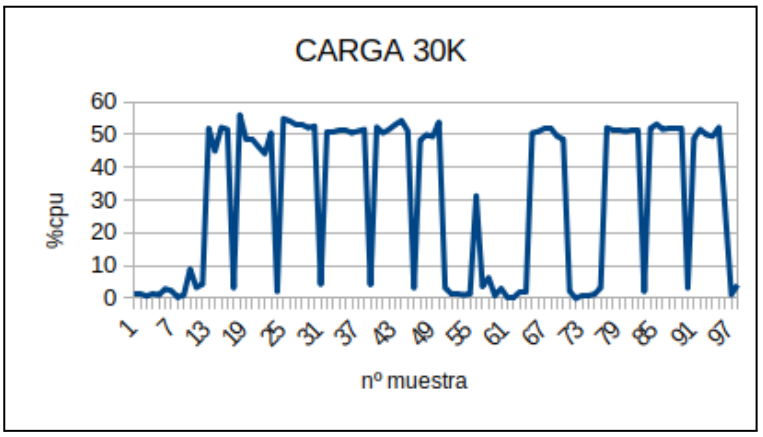
Producitivdad			
30K	60K	90K	120K
2391.04	940.77	298.5	191.9
2403.82	952.27	236.57	211.51
2365.06	953.08	237.83	192.14
2367.74	947.54	300.89	183.53
2291.48	868.94	268.97	184.08
2361.27	877.04	264.05	188.06
2346.61	942.64	293.95	248.08
2373.45	934.75	266.59	184.7
2358.49	949.72	524.44	206.48
2396.83	902.14	518.43	206.97
2345.29	945.29	464.16	204.49
2417.77	817.13	278.17	205.38
2337.11	947.12	370.33	194.1
2347.99	903.35	348.84	301.42
2239.34	812.93	295.97	352.34
2355.41	**910.45**	**310.6**	**209.71**

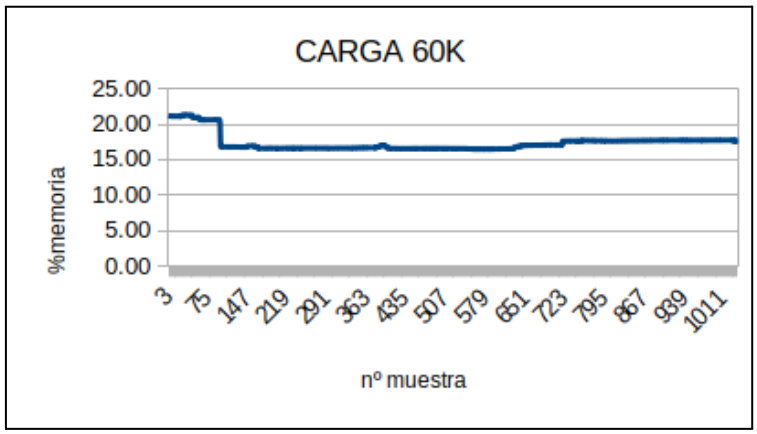
los números primos, y, por lo tanto, más largo es el tiempo de respuesta. Con el aumento de la carga, también se reduce la productividad, ya que esta se calcula dividiendo la carga entre el tiempo de respuesta. Como que el tiempo de respuesta va aumentando, la productividad tendrá un crecimiento inversamente proporcional.
Además, es importante destacar que el incremento en el tiempo de respuesta no sigue una progresión lineal. Pues la distribución de los números primos es irregular, lo que significa que por ejemplo, la cantidad de números primos entre 1 y 30 mil no es la mitad de la cantidad de números primos entre 1 y 60 mil.

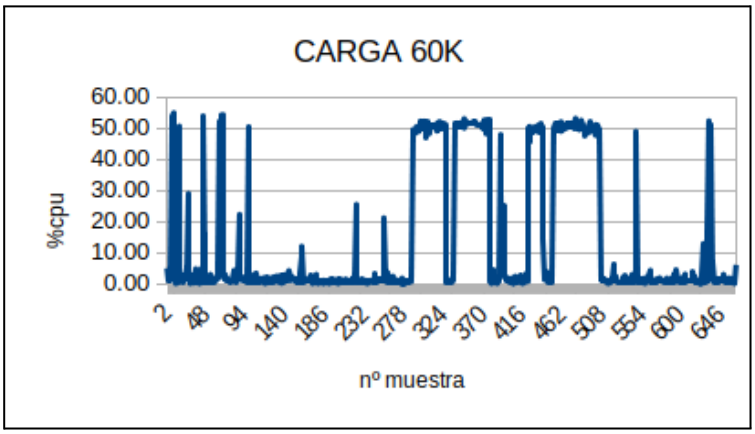
Pregunta 3

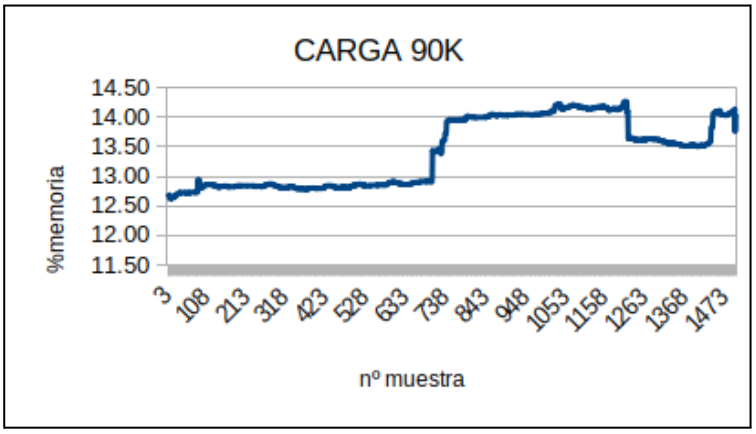
> ***¿Cuál es el porcentaje medio de uso de CPU y de memoria del sistema para cada una de las cargas ejecutadas? ¿Por qué se produce ese comportamiento? Además, muéstralo gráficamente a lo largo del tiempo de ejecución de la carga.***

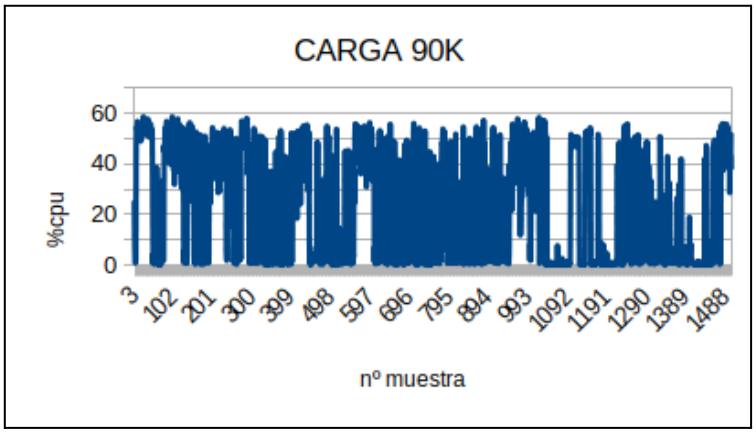

CARGA 30K
%memoria
25.90
25.80
25.70
25.60
25.50
25.40
25.30
1 8 15 22 29 36 43 50 57 64 71 78 85 92
nº muestra

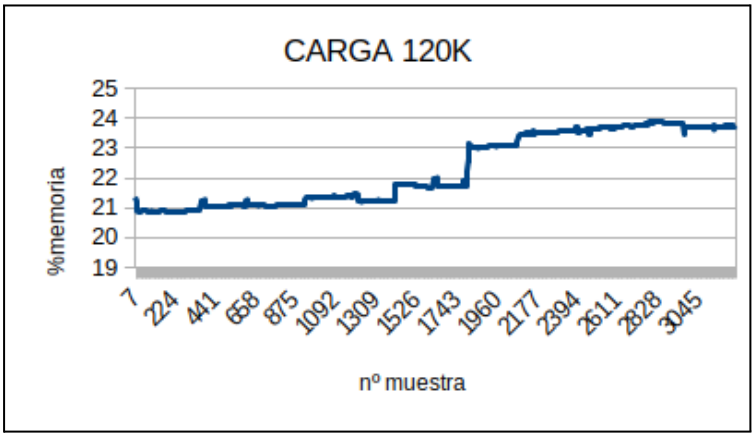

CARGA 30K
%cpu
60
50
40
30
20
10
0
1 7 13 19 25 31 37 43 49 55 61 67 73 79 85 91 97
nº muestra


CARGA 60K
%memoria
25.00
20.00
15.00
10.00
5.00
0.00
3 75 147 219 291 363 435 507 579 651 723 795 867 939 1011
nº muestra

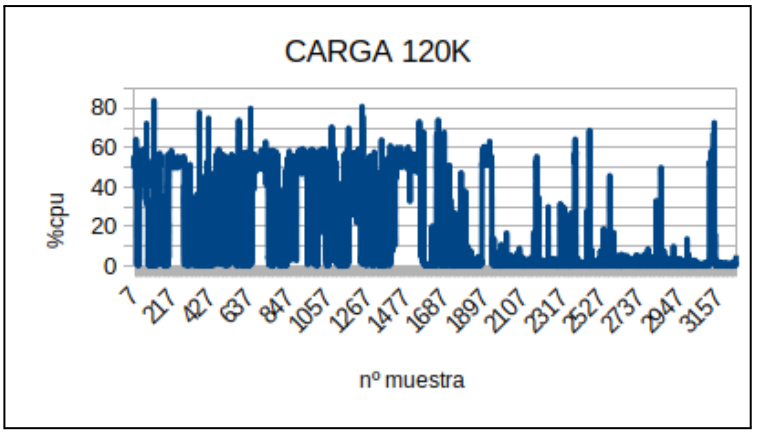

CARGA 60K
%cpu
60.00
50.00
40.00
30.00
20.00
10.00
0.00
2 48 94 140 186 232 278 324 370 416 462 508 554 600 646
nº muestra


CARGA 90K
%memoria
14.50
14.00
13.50
13.00
12.50
12.00
11.50
3 108 213 318 423 528 633 738 843 948 1053 1158 1263 1368 1473
nº muestra


CARGA 90K
%cpu
60
40
20
0
3 102 201 300 399 498 597 696 795 894 993 1092 1191 1290 1389 1488
nº muestra


CARGA 120K
%memoria
25
24
23
22
21
20
19
7 224 441 658 875 1092 1309 1526 1743 1960 2177 2394 2611 2828 3045
nº muestra

En lo que respecta al porcentaje de uso de la CPU, se observa que se mantiene casi constante alrededor del 50%, aunque en ocasiones alcanza valores muy bajos (menores al 10%). Probablemente, estos valores corresponden a los segundos de intervalo de muestreo (2 segundos). O también valores superiores a 50% sobre todo en la carga de 120 mil.



En cuanto al uso de memoria, este es relativamente bajo. Para la carga de 30,000, se sitúa en el 25%; para la carga de 60,000, aproximadamente en un 16% la mayor parte del tiempo; para la carga de 90,000, es aún más bajo, oscilando entre el 12.50% y el 14.40%; finalmente, para la carga de 120,000, el uso de memoria alcanza hasta el 24%.

A continuación, se deberá ejecutar la carga de trabajo igual a 90000 números primos, pero con el porcentaje de uso de la CPU al 100%. La experimentación se realizará de la misma forma que los casos anteriores (para el 50% de uso de la CPU). Si comparamos la carga de 90000 números primos con un 50% de CPU y un 100% de CPU, ¿qué diferencias hay en cuanto al tiempo de respuesta y la productividad?

Para alcanzar este objetivo, debemos emplear todos los hilos disponibles, es decir, 6 en total. Por lo tanto, el comando que ejecutaremos será el siguiente:
$sysbench cpu cpu --max-prime=<90000> --threads=<6> --time=0 --events=<90000> run

CARGA 90K (%CPU=100%)	
Tiempo de respuesta(s)	Productividad
153.9672	584.53
149.6334	601.46
150.0765	599.69
145.7849	617.34
133.5522	673.89
95.7867	939.58
93.1108	966.58
93.3296	964.32
95.034	947.02
93.5935	961.6
91.9483	978.8
253.8889	354.48
91.8666	979.67
94.7536	949.82
94.4625	952.75
122.05	**737.38**

La productividad de la carga de 90 mil con un 100% de uso de CPU es considerablemente mayor que la de la misma carga, pero con la mitad del uso de la CPU. Lo mismo puede decirse en relación con el tiempo de respuesta, el cual es significativamente menor cuando se utiliza el 100% de la CPU en comparación con el 50% de uso de CPU.

El tiempo de respuesta ha mejorado en un factor de aproximadamente 2.36 veces (288.393 / 122.05), y la productividad también (737.38 / 310.6 = 2.37).

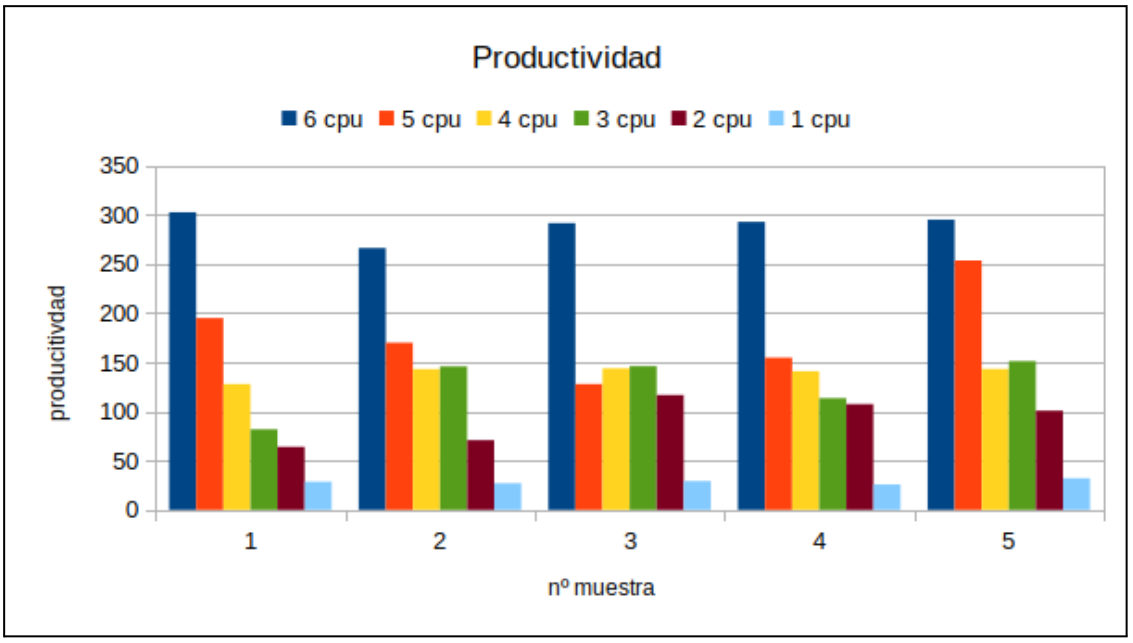
Para finalizar, se querrá evaluar el sistema actual cuando la carga es fija y se varían los recursos de la CPU (25% de CPU, 50% de CPU, ...). Para ello, se seleccionará la carga de 200000 números primos.

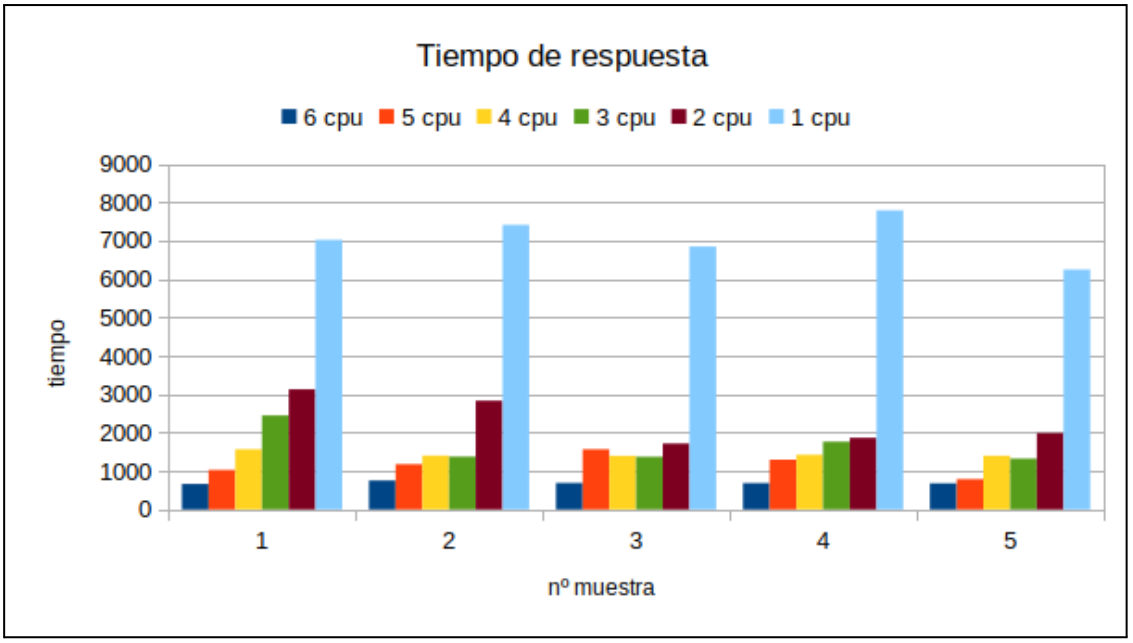
Pregunta 4

¿Cómo se comporta el tiempo de respuesta a medida que aumentan los recursos de la CPU? ¿Y la productividad?

Para este experimento, dada la carga relativamente alta, he decidido recopilar únicamente 5 muestras para cada porcentaje de CPU, con intervalos de 2 segundos entre cada una de ellas. Y como que el número de CPU de mi dispositivo no es divisible por $\frac{1}{4}$ ni por $\frac{3}{4}$ que corresponden respectivamente a los porcentajes 25% y 75% he optado para recoger muestras de todos los posibles valores de CPU del 1 hasta el 6.

Los resultados obtenidos se presentan en las siguientes gráficas:





Se observa que al aumentar la cantidad de CPU, disminuye el tiempo de respuesta, lo que a su vez aumenta la productividad. Además, como se mencionó anteriormente, el aumento en la productividad no sigue una tendencia lineal, sino que se aproxima a una distribución logarítmica, como se evidencia en estas gráficas.

Pregunta 5

¿Existe algún tipo de relación entre los recursos de la CPU y el tiempo de respuesta?

Sí, el aumento de los recursos de CPU conlleva un mejor rendimiento. Esto resulta bastante lógico, ya que con un mayor número de CPUs se pueden distribuir las tareas de manera equitativa entre ellas, lo que permite ejecutar múltiples tareas complejas en paralelo y tener mayor capacidad de procesamiento. Esto, a su vez, reduce el tiempo de respuesta y, en consecuencia, garantiza altos niveles de productividad.