**Práctica 1: Monitoreo y evaluación del desempeño de CPU, memoria y disco en Ubuntu Server**

**Ingeniería de software, FEI, UV**

**Agosto, 2025**

**Descripción**

Esta práctica tiene como objetivo **medir y comparar el desempeño del CPU, memoria y disco** de una instancia de **Ubuntu Server** utilizando programas de cálculo intensivo (generación de números primos) y operaciones de lectura/escritura en disco.

Se evaluarán diferentes escenarios:

* **Cálculo de números primos** implementado en **Java, C# y Python**, con y sin hilos.
* **Lectura y operaciones sobre un archivo de ~140 MB** generado con Python (20,000,000 líneas con dos enteros de tres cifras).
* **Monitoreo del sistema en tiempo real** mediante un programa en C.

**Objetivo**

* Comprender el impacto del **uso de hilos** en la eficiencia del CPU.
* Medir el rendimiento en **lenguajes de programación distintos** (Java, C# y Python).
* Analizar el efecto de operaciones intensivas en disco y memoria.
* Familiarizarse con el monitoreo de desempeño en **Linux Ubuntu Server**.
* Elaborar un **informe comparativo** con evidencias de pruebas, métricas y conclusiones.

**Metodología**

1. Preparar el entorno de trabajo en Ubuntu Server instalando **Java, C# y Python**.
2. Compilar y ejecutar programas que calculan números primos en los tres lenguajes, con y sin hilos.
3. Realizar pruebas **secuenciales (100 ejecuciones)** y **concurrentes (10 ejecuciones en paralelo con hilos)**.
4. Monitorear métricas del sistema (CPU, memoria, tiempo de ejecución) usando el programa monitor\_server.
5. Implementar un programa en Python que:
   * Genere un archivo con **20,000,000 líneas** de dos números de tres cifras cada uno (~140 MB).
   * Lea el archivo y sume los dos números por línea.
   * Mida el impacto en memoria, CPU y disco durante la operación.
6. Documentar las ejecuciones con **capturas de pantalla** y estadísticas recolectadas.
7. Redactar un informe final con los resultados y análisis.

**Elementos a considerar**

* Programa en **Java** para cálculo de primos con hilos y sin hilos.
* Programa en **C#** (Mono/.NET) para cálculo de primos con hilos y sin hilos.
* Programa en **Python** para cálculo de primos con hilos y sin hilos.
* Programa en **C** (monitor\_server) para monitoreo del desempeño del sistema.

**Etapas de la Práctica**

**1. Instalación de Java**

sudo apt update && sudo apt upgrade -y

sudo apt install default-jdk -y

java -version

**2. Instalación de C#**

Se puede trabajar con **Mono** para utilizar C#

sudo apt install -y mono-complete

mcs --version

mono --version

**\**2. Instalación de gcc***

Para instalar gcc utilice el siguiente comando

sudo apt install -y build-essential

gcc --version

**3. Instalación de Python**

En la mayoría de los casos, Python ya está instalado. Si no:

sudo apt install python3 -y

python3 --version

**4. Ejecución de Programas (Cálculo de Primos)**

**Compilación:**

# Java

javac CalculaPrimos.java

javac CalculaPrimosConHilos.java

# C#

mcs CalculaPrimos.cs

mcs CalculaPrimosConHilos.cs

# C (monitor del sistema)

gcc monitor\_server.c -o monitor\_server

**Ejecución básica:**

# Java

java CalculaPrimos

java CalculaPrimosConHilos

# C#

mono CalculaPrimos.exe

mono CalculaPrimosConHilos.exe

# Python

python3 CalculaPrimos.py

python3 CalculaPrimosConHilos.py

# Monitor del sistema

./monitor\_server

**5. Ejecución de pruebas**

Calibre el límite para que su instancia con 4 CPU tarde 10 segundos en ejectuar 1 tarea de cálculo de números primos con java.

Modifique la variable limite del método main en CalculaPrimos.java para lograr este objetivo

public static void main(String[] args) {

int limite = 10000000; // Puedes ajustar el límite aquí

CalculaPrimos calculador = new CalculaPrimos(limite);

calculador.run();

}

Compruebe el tiempo con

time java CalculaPrimos

El tiempo real debe ser de 10 segundos

**5.1 Ejecución secuencial (100 veces)**

# Java

for i in {1..100}; do java CalculaPrimos; done

# C#

for i in {1..100}; do mono CalculaPrimos.exe; done

# Python

for i in {1..100}; do python3 CalculaPrimos.py; done

**5.2 Ejecución concurrente (10 repeticiones con 10 hilos cada una)**

# Java con hilos

for i in {1..10}; do java CalculaPrimosConHilos & done; wait

# C# con hilos

for i in {1..10}; do mono CalculaPrimosConHilos.exe & done; wait

# Python con hilos

for i in {1..10}; do python3 CalculaPrimosConHilos.py & done; wait

⚠️ En todas estas pruebas, ejecutar ./monitor\_server en otra terminal para registrar **CPU, memoria y tiempos de ejecución**.

**6. Operaciones con archivo de ~140 MB**

* Crear un archivo con **20,000,000 líneas** (dos números de 3 cifras por línea):

import random

with open("datos.txt", "w") as f:

for \_ in range(20000000): # ~140 MB

a, b = random.randint(100,999), random.randint(100,999)

f.write(f"{a} {b}\n")

* Leer el archivo y sumar los números:

with open("datos.txt", "r") as f:

total = 0

for line in f:

a, b = map(int, line.split())

total += a + b

print(total)

* Monitorear el desempeño con ./monitor\_server.

**7. Evidencias**

* Capturas de pantalla de **uso de CPU, memoria y disco** durante las ejecuciones.
* Reportes estadísticos de métricas obtenidas con monitor\_server.
* Comparación entre ejecuciones secuenciales y concurrentes.

**Informe Final**

**Secciones:**

1. **Introducción**
   * Explicación del objetivo y contexto de la práctica.
   * Importancia del monitoreo de rendimiento.
2. **Metodología**
   * Pasos seguidos en la práctica.
   * Configuración de hardware y software.
   * Diagrama de stack de software con componentes y conexiones.
3. **Resultados**
   * Tablas y gráficas de consumo de CPU, memoria y tiempos de ejecución.
4. **Análisis**
   * Comparación entre ejecuciones en Java, C# y Python.
   * Diferencia entre programas con y sin hilos.
5. **Conclusiones**
   * Principales hallazgos.
   * Lenguaje más eficiente en el contexto probado.
   * Importancia del monitoreo.