Universidad Latina de Costa Rica

Ingeniería del Software

Sistemas Operativos II

Informe 1

Tema del informe:

Programación multi-procesador o
multi-core con Python o Javascript

Estudiante:

Isabela Méndez Zamora

Profesor:

Carlos Andres Mendez Rodriguez

09 de febrero del 2024

San Pedro, Montes de Oca

Informe

Es importante para hablar de estos tipos de programación entender que en el caso de Python es un multiprogramador y multiprocesador de tareas que nos permite crear códigos más eficientes pero también con un menor tiempo de ejecución, lo cual podemos conseguir intercalando las tareas (simultáneamente) o en paralelo. La programación multiprocesador o multicore se refiere a la capacidad de un sistema para ejecutar múltiples procesos o tareas de forma simultánea, aprovechando los múltiples núcleos de procesamiento disponibles en una CPU. Este enfoque permite mejorar el rendimiento y la eficiencia de las aplicaciones al distribuir la carga de trabajo entre varios núcleos.

En la programación multiprocesador, es fundamental comprender la diferencia entre la ejecución sincrónica y la asincrónica:

- **Ejecución Sincrónica:** En este enfoque, las tareas se ejecutan secuencialmente, una tras otra. Cada tarea debe completarse antes de pasar a la siguiente. Esto puede limitar el rendimiento, ya que los recursos de la CPU pueden quedar subutilizados mientras se espera que una tarea termine.
- **Ejecución Asincrónica:** En contraste, la ejecución asincrónica permite que múltiples tareas se ejecuten simultáneamente. Las tareas pueden iniciar, detenerse y continuar en cualquier momento, lo que aprovecha al máximo los recursos del sistema y mejora la eficiencia.

En Python, el módulo multiprocessing proporciona soporte para la programación multiprocesador. Puede crear procesos independientes que se ejecuten simultáneamente, lo que permite aprovechar los múltiples núcleos de la CPU. En JavaScript, el modelo de ejecución es principalmente asincrónico debido a su naturaleza basada en eventos y su bucle de eventos (event loop). Sin embargo, con la introducción de Web Workers, también se puede lograr la ejecución sincrónica. La programación multi procesador/multicore es esencial para mejorar el rendimiento y la eficiencia de las aplicaciones al aprovechar al máximo los recursos del sistema. Tanto en Python como en JavaScript, se pueden implementar soluciones tanto sincrónicas como asincrónicas para aprovechar al máximo los

múltiples núcleos de la CPU. Es crucial comprender las diferencias entre la ejecución sincrónica y asincrónica para diseñar sistemas eficientes y escalables.

Ejemplos

- **Asincrónico en Python**

```
import multiprocessing



def tarea(num):

    print(f'Proceso {num} ejecutándose')



if __name__ == '__main__':

    procesos = []

    for i in range(5):

        p = multiprocessing.Process(target=tarea, args=(i,))

        procesos.append(p)

        p.start()



    for p in procesos:

        p.join()
```

- **Asincrónico en JavaScript**

```
function tarea(num) {
 console.log(`Proceso ${num} ejecutándose`);
}

for (let i = 0; i < 5; i++) {
 setTimeout(() => tarea(i), 0);
}
```

La programación multi procesador/multicore permite mejorar el rendimiento y la capacidad de respuesta de las aplicaciones al distribuir la carga de trabajo entre múltiples núcleos de CPU.

- En Python, se puede implementar programación multiprocesador utilizando multithreading o multiprocessing. La biblioteca threading se utiliza para el multithreading, mientras que la multiprocessing se utiliza para el multiprocesamiento. El módulo multiprocessing en Python permite crear procesos independientes que se ejecutan simultáneamente. Proporciona una interfaz similar a la del módulo threading, pero utiliza procesos en lugar de subprocesos. En Python, el GIL puede ser un limitante en el rendimiento de las aplicaciones multithreaded, ya que solo permite que un hilo ejecute código de Python a la vez. Sin embargo, esto no afecta a la programación multiprocesador con el módulo multiprocessing, ya que cada proceso tiene su propio intérprete y espacio de memoria.

- En JavaScript, los Web Workers permiten ejecutar scripts en segundo plano en hilos separados, lo que permite el paralelismo en el navegador web.Aunque JavaScript es principalmente un lenguaje asincrónico debido a su modelo de ejecución basado en eventos y el bucle de eventos, los Web Workers permiten introducir paralelismo en las aplicaciones web.