Tarea 1
Computación Concurrente

Macias Gomez Jorge

October 2020

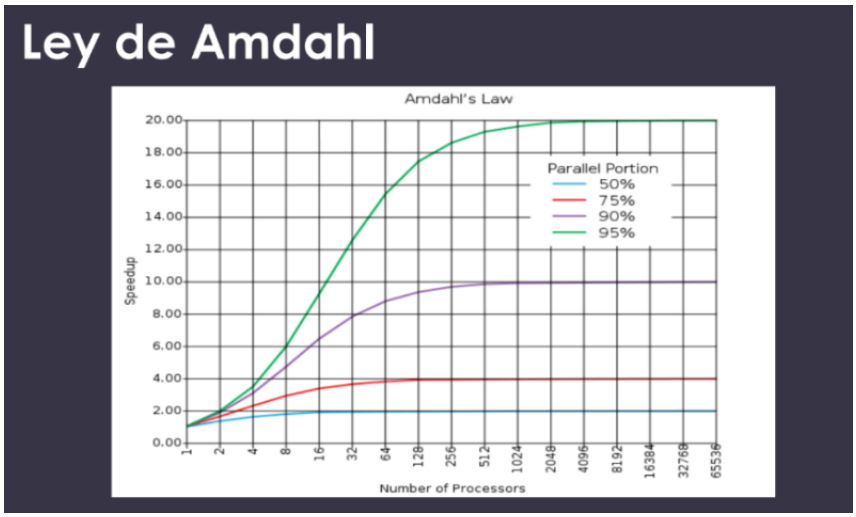
2.

- fork : El método funciona para crear un nuevo proceso hijo.
- exit : Terminar el proceso.
- wait : La llamada al sistema suspende la ejecución del proceso actual hasta que un proceso hijo haya salido o hasta que se haya entregado una señal cuya acción sea terminar el proceso actual.
- getpid : Regresa el id del proceso.
- getppid : Regresa el id del proceso padre.
- getuid : Regresa el id del usuario.
- pid_t : Identificador real del proceso actual.
- pthread_create : crea un hilo (análogo a fork).
- pthread_join : Espera a que acabe el hilo.
- pthread_exit : Termina un hilo.
- pipe: Un pipe es una relación entre 2 procesos, tal que el output de un proceso será el input del otro proceso.
- close : Termina un proceso.

3.

- Global Interpreter Lock (GIL): es el mecanismo utilizado en CPython para impedir que múltiples threads modifiquen los objetos de Python a la vez en una aplicación multihilo.

- Ley de Amdahl. Dice que el el speedup que se puede alcanzar al volver concurrente un programa esta dado por:
  $speedup = \frac{\text{tiempo de ejecución secuencial}}{\text{tiempo de ejecución paralela}}$
  También menciona que el máximo speedup alcanzable por n procesos esta limitado por la fracción p del programa que puede ser paralelizable:
  $speedup(n) = \frac{1}{1-p+\frac{p}{n}}$
  tal que (1-p) es la parte secuencial y $\frac{p}{n}$ la parte paralela



- multiprocessing. Es un paquete que admite generar procesos mediante una API similar al módulo de subprocesos. El paquete de multiprocesamiento ofrece simultaneidad local y remota, evitando efectivamente el bloqueo de intérprete global mediante el uso de subprocesos en lugar de hilos.
  - Pipe() devuelve un par de objetos de conexión conectados por un pipe bidireccional.
  - active_children(). Devuelve la lista de todos los hijos vivos del proceso. actual.
  - current_process(). Devuelve el objeto de proceso correspondiente al proceso actual.
  - cpu_count(). Devuelve el número de CPU del sistema.
  - Manager(). Devuelve un objeto SyncManager iniciado que se puede utilizar para compartir objetos entre procesos.

4 multiprocessing.process

- Multiprocesamiento funciona creando un objeto del process y luego llamando a su método start() .

- Primero Importamos la clase de proceso y luego instanciar el objeto del proceso con la función de saludo que queremos ejecutar.
- Luego contamos el proceso para comenzar a usar el método start(), y finalmente terminamos el proceso con el método join().

```
from multiprocessing import Process


def square(x):

    for x in numbers:
        print('%s squared  is  %s' % (x, x**2))


def is_even(x):

    for x in numbers:
        if x % 2 == 0:
            print('%s is an even number ' % (x))


if __name__ == '__main__':
    numbers = [43, 50, 5, 98, 34, 35]

    p1 = Process(target=square, args=('x',))
    p2 = Process(target=is_even, args=('x',))

    p1.start()
    p2.start()

    p1.join()
    p2.join()

    print "Done"
```