UNIVERSIDAD TECNOLÓGICA DE SANTIAGO (UTESA)

Área de Arquitectura e Ingeniería
Carrera de Ingeniería en sistemas computacionales.

ALGORITMOS PARALELOS

Tarea Semana 1: Realizar la investigación y realizar una presentación a la clase.

PRESENTADO POR:

José Rodolfo Morel. 1-16-0328.

ASESOR:

Ing. Iván Mendoza.

GRUPO:

INF-025-001

Santiago De Los Caballeros
República Dominicana
Mayo, 2021.

Realizar la investigación y realizar una presentación a la clase.

Investigar:

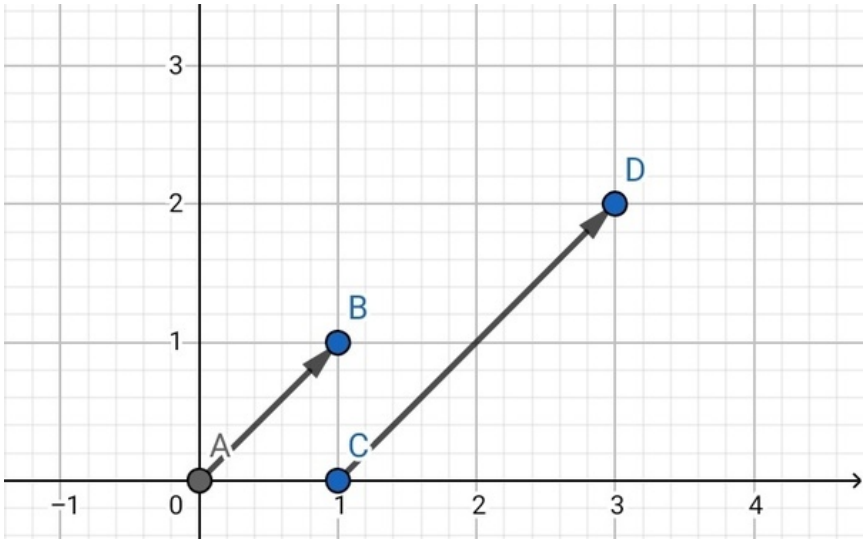
- Algoritmos

un algoritmo es simplemente un conjunto bien definido de instrucciones que indican cómo realizar una operación específica o solucionar un problema determinado.

Un ejemplo de un algoritmo, sería una receta de cocina, o el manual de instalación de algún equipo o electrodoméstico, entre otras cosas.

- Paralelo

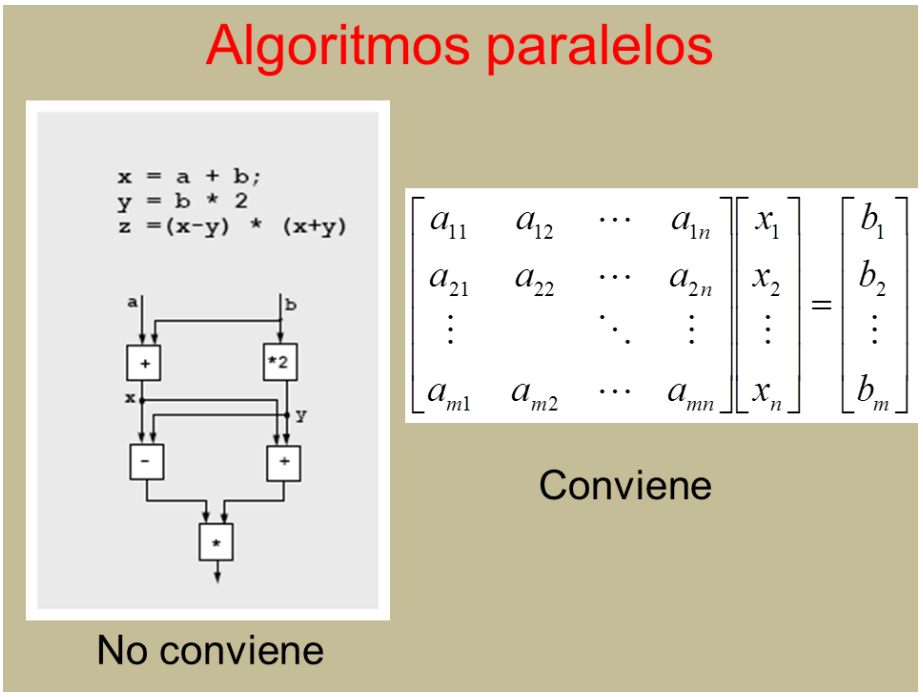
Paralelo se refiere a un objeto o entidad que existe próximo o al lado de otra, por ejemplo podemos tener procesos o actividades que pueden existir o realizarse al mismo tiempo.



- Algoritmos paralelos

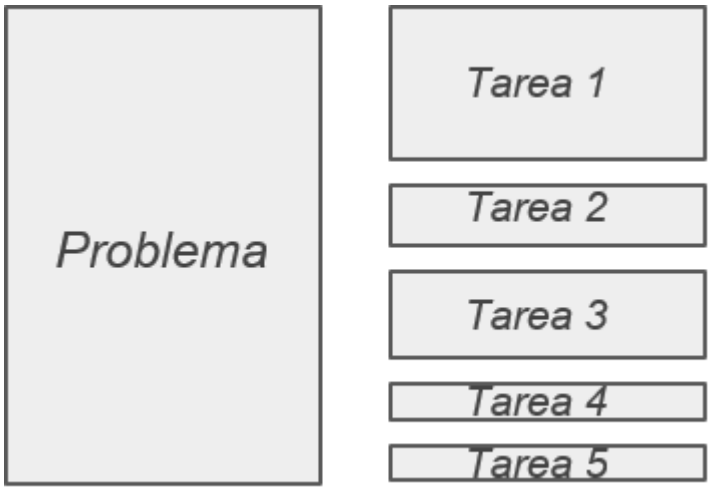
Normalmente en un algoritmo convencional un conjunto de instrucciones se ejecutan de manera secuencial en un único procesador una después de otra, pero en los algoritmos paralelos las instrucciones se separan en varias partes que son ejecutadas de forma simultánea en diferentes procesadores.

Debido a esto los algoritmos paralelos representan una enorme ventaja de rendimiento comparado con los algoritmos secuenciales. Por ejemplo, si un algoritmo que se conforma de 5 instrucciones que tardan 5 segundos en ejecutarse cada una, se ejecuta de forma secuencial el algoritmo tarda 25 segundos en ejecutarse, mientras en paralelo cada instrucción se ejecutará al mismo tiempo por lo que el tiempo sería de solo 5 segundos en total.



- Programación paralela

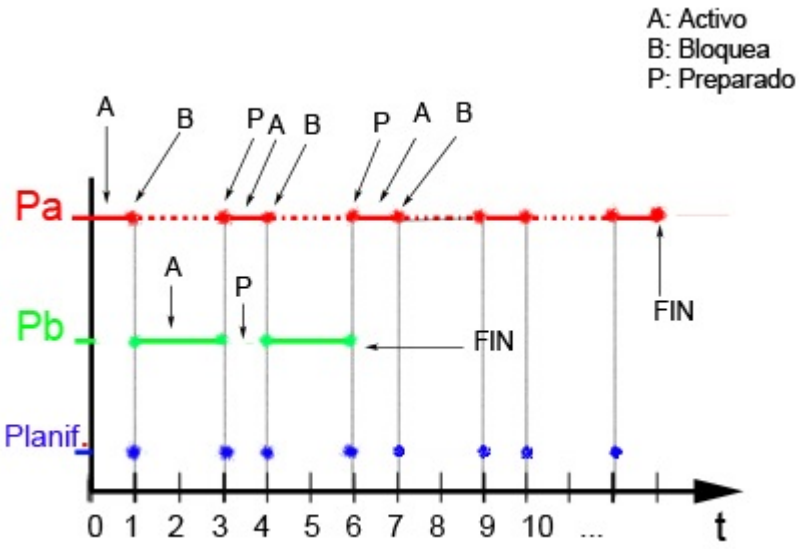
Se trata de un paradigma de programación que se especializa en la escritura de programas que se ejecutan de forma paralela.



La programación paralela se enfoca en dividir las instrucciones que se desean ejecutar en varias partes y ejecutarlas en diferentes hilos para acelerar la ejecución del programa o para ayudar a resolver problemas más complejos que no se podrían resolver de forma normal.

- Programación concurrente

Muchas veces la concurrencia se confunde con el paralelismo, mientras que el paralelismo varias tareas se ejecutan al mismo tiempo en procesadores separados, en la concurrencia se ejecutan varias tareas a la vez pero no al mismo tiempo pero en un mismo procesador, pero en diferentes hilos de tiempo, un ejemplo es la imagen de abajo.

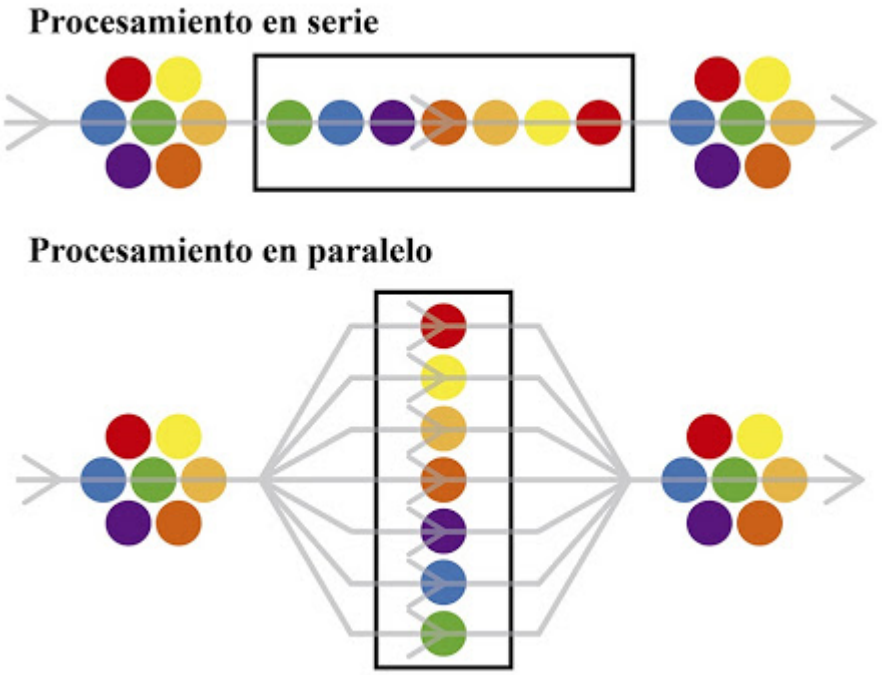


Como podemos ver en la foto hay varias tareas ejecutándose a la vez pero cada una ocupa diferentes momentos en el tiempo, como podemos ver las 3 tareas nunca coinciden a la vez en algún momento del tiempo.

En la concurrencia no sería correcto decir que las tareas se ejecutan al mismo tiempo, porque como vimos no es así, sin embargo la importancia de la concurrencia yace en la capacidad de hacer que el procesador alterner entre varias tareas de forma tan rápida que parezca que estas se ejecuten al mismo tiempo.

- Paralelismo

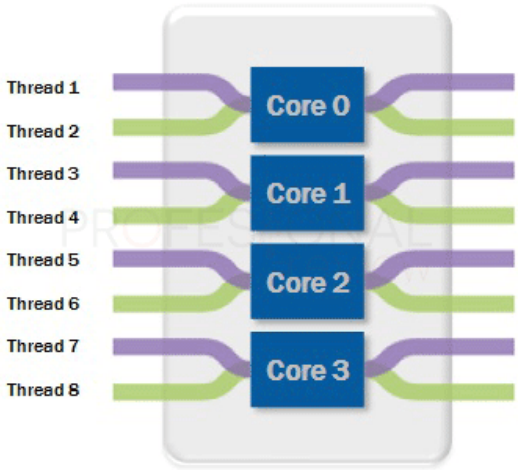
En esta área el paralelismo se refiere a tareas o actividades que se ejecutan simultáneamente, como podemos ver en la imagen siguiente:



La concurrencia es similar a una persona realizando 3 tareas al mismo tiempo, alternando entre cada una, el paralelismo sería equivalente a tener 3 personas cada una realizando cada una de estas tareas.

- Hilos

A nivel de procesador los hilos no existen físicamente ya que se trata de una pila lógica de instrucciones a las cuales el procesador les asigna una prioridad de ejecución en un tiempo dado.



Los hilos son la forma como los procesadores agrupan diferentes instrucciones para después ejecutarlas posteriormente.

- Lenguajes o Frameworks de programación que usan paralelismo o programación concurrente

- ☐ **CUDA** (usado por nvidia para acelerar el procesamiento de gráficos)
- ☐ **APACHE HADOOP** (Framework usado en servidores)
- ☐ **OPENMP** (Lenguaje de creación de API y bibliotecas de recursos compartidos en paralelo importables en FORTRAN y C/C++)
- ☐ **MPI** (Message Passing Interface) (Es un lenguaje paralelo que permite la creación de un interface de comunicación entre las diferentes rutinas en ejecución)
- ☐ **TITANIUM** (Lenguaje paralelo con un dialecto basado en JAVA, enfocada a la programación en múltiples procesadores)
- ☐ **COLECCIÓN CONCURRENTE (CNC)** (Lenguaje diseñado para el procesamiento de grandes cantidades de datos en forma de tuplas o registros de forma concurrente)

Ejercicios

Ejemplos en Código que permita ejecutar tareas en paralelo, concurrencia o paralelismo en cualquier lenguaje y explicar su código. (Al menos 5 Código)

Ejercicio 1:

```go
package main

import ("fmt")

//Función que imprime un texto pasado por parámetro
func print(str string) { fmt.Println(str)}

//Método principal
func main() {
    //Imprime un grupo de mensajes de forma asíncrona
    go print("Hola 1")
    go print("Hola 2")
    go print("Hola 3")
    go print("Hola 4")
    go print("Hola 5")
    go print("Hola 6")
```

```
//Hago un scan para bloaquear el programa y este no finalize
hasta que el usuario inserte algun caracter
    var wait string
    fmt.Scanln(&wait)
}
```

Ejercicio 2:

```
package main

import ("fmt" "sync")

//Declaramos la cola global para poder acceder a ella desde
diferentes metodos
var cola sync.WaitGroup

//Metodo que imprime una secuencia de mensajes n veces
func rutina(str string) {
    for i := 0; i < 3; i++ {
        fmt.Println(str, i)
    }
    //Notifico a la cola que esta rutina término
    cola.Done()
}

//Metodo principal
func main() {

    //Le indicamos a la cola que debe esperar 3 rutinas
    cola.Add(3)

    //Ejecuta 3 tareas de forma concurrente pero una vez que
una de ellas inicia las otras se bloan hasta que esta termine
imprimiendo un mensaje desde el punto de control de cada
rutina

    go rutina("rutina 1: ")
    go rutina("rutina 2: ")
    go rutina("rutina 3: ")

    //Bloquea el programa para que este no finaliza hasta que
el método Done() sea llamado 3 veces
    cola.Wait()
}
```

Ejercicio 3:

```
package main

import ("fmt")

func print(str string) { fmt.Println(str) }

func sumarRango(desde int, hasta int) int {

    var suma int

    for i := desde; i < hasta; i++ {
        fmt.Println("de ", desde, "a", hasta, " => ", suma, "
+ ", i, " = ", (suma + i))
        suma = suma + i
    }
    return suma
}

//Metodo principal
func main() {

//Realiza la suma de un rango de numeros desde x hasta y, para
esto divide el rango en 4 partes las cuales se suman de forma
separadas en diferentes rutinas para luego unir los resultados
agilizando asi por 4 la velocidad en la realizacion de dicha
suma.

    var desde int
    var hasta int

    var total int

    //Le pido al usuario los rangos desde y hasta
    print("Ingrese el rango de la suma concurrente. ")
    print(" desde => ")
    fmt.Scanln(&desde)
    print(" hasta => ")
    fmt.Scanln(&hasta)

    //Divido el total de numeros dentro del rango en 4 partes
para sumarlas por separado
    offset := (hasta - desde) / 4

    //Realizo la sumatoria en diferentes rangos, por ejemplo
si el rango es de 1 a 100, realizo la suma en 4 rangos de a 25
numeros cada uno en diferentes rutinas para acelerar el tiempo
x 4.
```

```
    go func() {
        total = total + sumarRango(desde, desde+offset)
    }()

    go func() {
        total = total + sumarRango(desde+offset,
desde+(offset*2))
    }()

    go func() {
        total = total + sumarRango(desde+(offset*2),
desde+(offset*3))
    }()

    go func() {
        total = total + sumarRango(desde+(offset*3), hasta)
    }()

    //Hago un scan para bloaquear el programa y este no
finalize hasta que el usuario inserte algun caracter
    var wait string
    fmt.Scanln(&wait)

    //Imprimo el total de la sumatoria
    fmt.Println("total = ", total)
}
```

Ejercicio 4:

```
package main

import ( "fmt" "math")

//Funcion que imprime un texto pasado por parametro
func print(str string) {  fmt.Println(str) }

//Variable global compartida entre rutinas que almacenara el
mayor numero encontrado
var mayor int

//Esta funcion recibe un array y busca desde la izquierda
hacia la derecha el mayor numero y se detiene en el centro.
func buscarIzq(array [10]int) {

    print("Buscando de izquierda a derecha")
    var hasta int
    hasta = int(math.Round(float64(len(array) / 2)))
```

```
for i := 0; i < hasta; i++ {
        if array[i] > mayor {
            fmt.Println("Desde la Izquierda  =>  ", array[i],
"mayor que ", mayor)
            mayor = array[i]
        }
    }
}

//Esta funcion recibe un array y busca desde la derecha hacia
la izquierda el mayor numero y se detiene en el centro.
func buscarDer(array [10]int) {

    print("Buscando de derecha a izquierda")
    var hasta int
    hasta = int(math.Round(float64(len(array) / 2)))

    for i := (len(array) - 1); i > hasta; i-- {
        if array[i] > mayor {
            fmt.Println("Desde la Derecha  =>  ", array[i],
"mayor que ", mayor)
            mayor = array[i]
        }
    }
}

//Metodo principal
func main() {

    //buscar el mayor numero en el arreglo para esto lo divide
en 2 partes y comeinza a recorrerlo por izquierda y derecha
hasta terminar en la mitad
    print("Buscando el mayor en: ")
    array := [10]int{2, 3, 6, 7, 8, 1, 23, 34, 4, 5}
    fmt.Println(array)

    //busco el mayor recorriendo la parte derecha y izquierda
del array al mismo tiempo
    go buscarIzq(array)
    go buscarDer(array)

    //Hago un scan para bloquear el programa y este no
finalize hasta que el usuario inserte algun caracter
    var wait string
    fmt.Scanln(&wait)

    //Imprimo el mayor
    fmt.Println("El mayor es = ", mayor)
}
```

Ejercicio 5:

```go
package main

import (
    "fmt"
    "math/rand"
    "sync"
)

//Funcion que imprime un texto pasado por parametro
func print(str string) {
    fmt.Println(str)
}

//Declaramos la cola global para poder acceder a ella desde
diferentes metodos
var cola sync.WaitGroup

//Funcion que genera un numero al azar dentro del rango
especificado.
func generarRandon(posicion string, rango int) {
    fmt.Println(posicion, rand.Intn(rango))
    cola.Done()
}

//Metodo principal
func main() {

    //generar cada numero del loto de forma asincrona usando
numeros randon

    fmt.Println("Los 7 numeros ganadores son: ")
    //Le indicamos a la cola que debe esperar 7 rutinas
    cola.Add(7)

    go generarRandon("Primer numero => ", 38)
    go generarRandon("segundo numero => ", 38)
    go generarRandon("tercero numero => ", 38)
    go generarRandon("cuarto numero => ", 38)
    go generarRandon("quinto numero => ", 38)
    go generarRandon("sexto numero => ", 38)

    go generarRandon("loto mas => ", 10)

    //Bloquea el programa para que este no finalize hasta que
el metodo Done() sea llamado 7 veces
    cola.Wait()

}
```

Ejercicios disponibles en:

Repositorio:
https://github.com/RodolfoMH/algoritmos_paralelos_tareas.git