Moises Vidal Hernandez S5

**Programación paralela**

En cuestiones de la demanda continua del poder computacional específicamente en las áreas importantes como la predicción de meteorología, biocomputación y astrofísica, para dar una solución a esta demanda se emplea la programación paralela.

Tenemos que decir que la programación paralela es el uso simultáneo de múltiples recursos computacionales para resolver un problema computacional, también las computadoras secuenciales convencionales han venido aumentando considerablemente su velocidad, aunque no en relación con sistemas cada vez más complejos que requieren mayor tiempo de cómputo.

Algunos a entender la programación paralela son los siguientes

* Diseño de Computadoras paralelas.
* Diseño de [Algoritmo](https://www.ecured.cu/Algoritmo) eficiente.
* Métodos para evaluar el Algoritmo paralelo.
* Lenguajes para computadores paralelos, flexibles para permitir una implementación eficiente y que sean fáciles de programar.
* Herramientas para la programación paralela.
* Programa Paralelo portable.
* [Compilador Paralelizarte](https://www.ecured.cu/Compilador_Paralelizante).

**Ventajas y desventajas de la programación paralela**

Ventajas

* Resuelve problemas que no se podrían realizar en una sola CPU
* Resuelve problemas que no se pueden resolver en un tiempo razonable
* Permite ejecutar problemas de un orden y complejidad mayor
* Permite ejecutar código de manera más rápida (aceleración)
* Permite ejecutar en general más problemas
* Obtención de resultados en menos tiempo
* Permite la ejecución de varias instrucciones en simultáneo
* Permite dividir una tarea en partes independientes.

Desventajas

* Mayor consumo de energía
* Mayor dificultad a la hora de escribir programas
* Dificultad para lograr una buena sincronización y comunicación entre las tareas
* Retardos ocasionados por comunicación ente tareas
* Número de componentes usados es directamente proporcional a los fallos potenciales
* Condiciones de carrera
  + Múltiples procesos se encuentran en condición de carrera si el resultado de los mismos depende del orden de su llegada
  + Si los procesos que están en condición de carrera no son correctamente sincronizados, puede producirse una corrupción de datos

**Tipos de paralelismo**

* Paralelismo a nivel de bit:

Se habla de paralelismo al nivel de bit, cuando se aumenta el tamaño de la palabra del procesador (tamaño de la cadena de bits a procesar). Este aumento reduce el número de instrucciones que tiene que ejecutar el procesador en variables cuyos tamaños sean mayores a la longitud de la cadena.

Ejemplo: En un procesador de 8-bits sumar dos números de 16bits tomaría dos instrucciones. En un procesador de 16-bits esa operación requiere solo una instrucción.

Nota: este método está “estancado” desde el establecimiento de las arquitecturas de 32 y 64 bits.

* Paralelismo a nivel de instrucción

Este tipo de paralelismo consiste en cambiar el orden de las intrucciones de un programa y juntarlas en grupos para posteriormente ser ejecutados en paralelo sin alterar el resultado final del programa.

Pipelining

El pipelining proviene de la idea de que en una tubería no es necesario esperar a que todo el agua dentro salga, para que pueda entrar más. Los procesadores modernos tienen un 'pipeline' que separa las instrucciones en varias etapas, donde cada etapa corresponde a una acción diferente que necesita la salida de la anterior.

Ejemplo: Un pipeline de 5 etapas: fetch (buscar la instrucción), decode (decodificarla), execute (ejecutarla), write (escribir en memoria el resultado de la operación).

* Paralelismo a nivel de datos

Cada procesador realiza la misma tarea sobre un subconjunto independiente de datos.

Ej: Dos granjeros se dividen el área de césped a podar.

El caso clásico de paralelismo de datos, es el cálculo de pi por partes usando el método de monte carlo

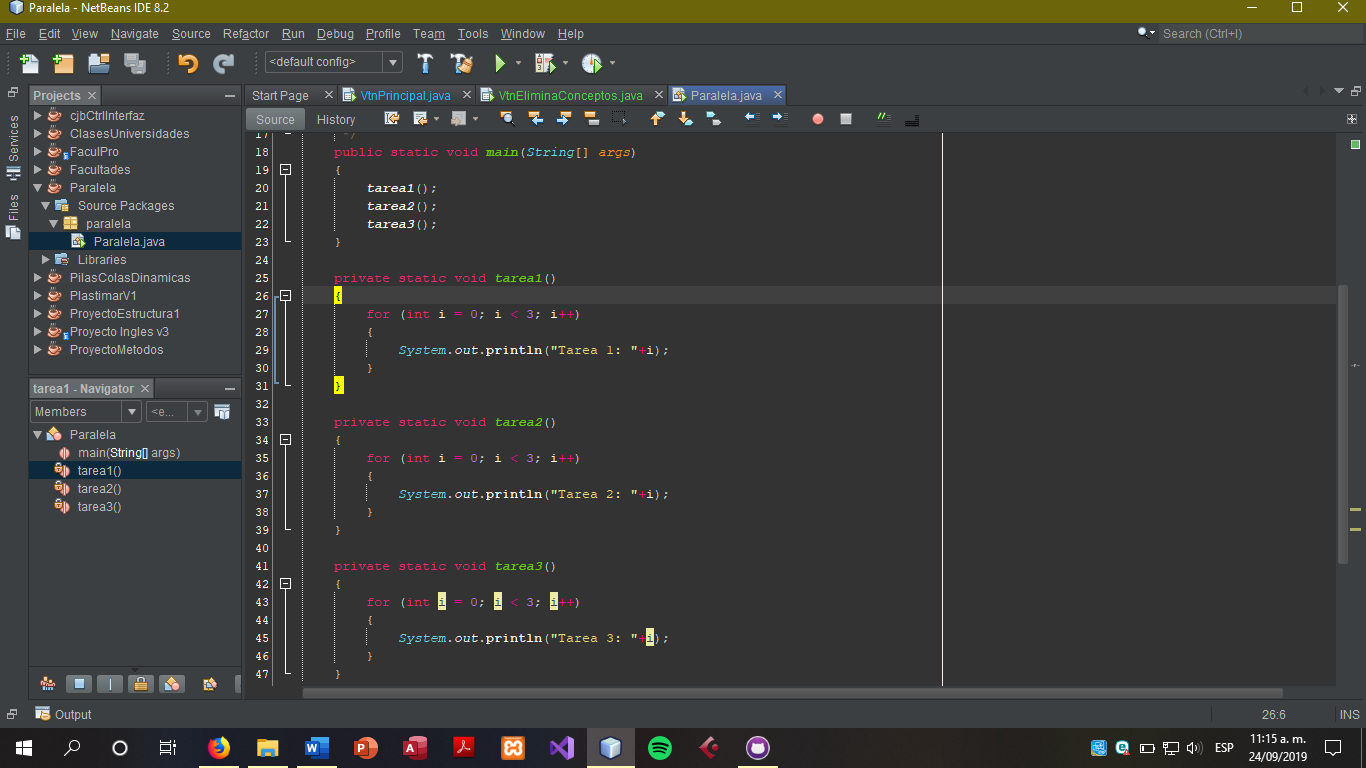
* Paralelismo a nivel de tareas

Cada hilo realiza una tarea distinta e independiente de las demás.

Ej: Un granjero poda el césped, el otro cosecha.

**Ejemplo**

Tomaremos un ejemplo de programación secuencial donde se van a ejecutar tres tareas de forma secuencial. Estas tareas son muy sencillas: consisten en un **ciclo for** que va desde 0 hasta 2, imprimiendo el valor del contador en cada ciclo.



Una vez ejecutado el código , podemos ver que las tareas se ejecutan una detars de la otra de la siguiente forma

* Primero los 3 ciclos de la tarea1
* Luego los 3 ciclos de la tarea2
* Y por ultimo los 3 ciclos de la tarea3

La salida es la siguiente

