## Memoria CPA

López Muñoz, Alejandro

Benites Aldaz, Dairon Andrés

A green circuit board

Description automatically generated with low confidence

### Índice

[Ejercicio 1 3](#_Toc88844666)

[Ejercicio 2 4](#_Toc88844667)

[Ejercicio 3 5](#_Toc88844668)

[Ejercicio 4 6](#_Toc88844669)

[Ejercicio 5 7](#_Toc88844670)

[Ejercicio 6 8](#_Toc88844671)

[Anexos 9](#_Toc88844672)

# Ejercicio 0

Text

Description automatically generated

# Ejercicio 1

**Obtén una primera versión paralela del programa paralelizando las funciones que utiliza la función process: *distance* y *swap*. Llama a esta versión restore1.c**

***distance***

Hemos decidido paralelizar el bucle for, para ello ha sido necesario declarar “private” las variables *j, r,g,b* ya que en cada iteración se producen modificaciones de las variables, por tanto podrían ocurrir conflictos entre ejecuciones de distintos hilos.

Hemos declarado *reduction(+:d)*  ya que necesitamos la suma de todas las iteraciones de los hilos.

Text

Description automatically generatedText

Description automatically generated

***swap***

Hemos decidido paralelizar el bucle for, para ello ha sido necesario declarar “private” las variables *x,d y aux* ya que en cada iteración se producen modificaciones de las variables, por tanto podrían ocurrir conflictos entre ejecuciones de distintos hilos.

# Ejercicio 2

**Obten una segunda versión paralela paralelizando el cuerpo de la función process, dejando la función swap paralelizada (pero no la función distance). Observa que los bucles externos de la función process no se pueden paralelizar, con lo que deberás paralelizar los bucles internos. Llama a esta versión restore2.c.**

Text

Description automatically generated

Hemos decidido paralelizar los dos bucles for [x2,y2], ya que los bucles externos no se pueden paralelizar.

Para ello ha sido necesario declarar “private” la variable *d* ya que en cada iteración se producen modificaciones de la variable, por tanto, podrían ocurrir conflictos entre ejecuciones de distintos hilos.

Ha sido necesario declarar una sección crítica en la actualización de la variable *min* y *my/mx.*

# Ejercicio 3

**En este ejercicio no se pide que modifiques ningún código ni que realices ninguna ejecución, sino que respondas de forma razonada a lo que se pregunta.**

**En la paralelización del bucle y2 de process, ¿crees que con alguna planificación se obtendría un mejor equilibrio de carga que con otra? ¿Sí/no? ¿Cuáles? ¿Por qué?**

y2 -> Tiene una carga equilibrada por tanto no hace falta una planificación diferente.

**¿Y en el bucle y de process? (como se ha dicho, ese bucle no se puede paralelizar, pero para este ejercicio teórico supongamos que sí se puede).**

y -> Sí que tiene un desequilibrio de cargas por tanto no sería una buena opción la planificación por defecto

# Ejercicio 4

**Utilizando los nodos de cálculo del cluster kahan, saca tiempos de ejecución de las dos versiones paralelas realizadas, usando 16 hilos y las siguientes planificaciones:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Static con tamaño de *chunk* por defecto | static con tamaño de *chunk* 1. | Dynamic con tamaño de chunk por defecto |
| restore1.c | **17.974302** | **28.464924** | **199.36364536** |
| restore2.c | **5.7226924** | **8.768748** | **8.4352** |

**Chart, scatter chart

Description automatically generated  
Analiza cuál es la mejor planificación en cada versión paralela, indicando a qué puede deberse.**

Como es evidente la mejor planificación definida es la ***Static con chunk por defecto*** ya que ***la*** carga es equilibrada entre los distintos hilos de ejecución.

Esto se debe a … {/\*Completar\*/}

# Ejercicio 5

**Utilizando los nodos de cálculo del cluster kahan, saca tiempos de ejecución de las dos versiones paralelas realizadas, variando el número de hilos y eligiendo en cada versión la planificación con la que se hayan obtenido mejores resultados en el ejercicio anterior. Para limitar el número de ejecuciones, se recomienda usar potencias de 2 para los valores del número de hilos (2, 4, 8...), llegando hasta el número de hilos que consideres adecuado (justifica por qué eliges ese número máximo de hilos).**

**Muestra tablas y gráficas para tiempos, speed-ups y eficiencias para las versiones paralelas que has realizado. Utiliza esas tablas y gráficas para comparar las prestaciones de las dos versiones. En vista de los resultados, extrae conclusiones sobre cuál es la mejor versión paralela, o bien si no hay diferencia significativa, y razona a qué crees que se debe.**

**Explica cómo has lanzado las ejecuciones en el cluster, indicando cómo estableces el número de hilos y la planificación y adjuntando alguno de los ficheros de trabajo utilizados.**

|  |  |  |
| --- | --- | --- |
| N\_Threads | restore1.c | restore2.c |
| 1 | 59.908 | 59.617 |
| 2 | 32.468 | 29.951 |
| 4 | 19.994 | 15.334 |
| 8 | 17.077 | 8.288 |
| 16 | 18.226 | 5.481 |
| 32 | 23.911 | 3.030 |
| 64 | 41.974 | 2.453 |

**Chart, line chart

Description automatically generated**

|  |  |
| --- | --- |
| restore1.c | restore2.c |
| *N\_Threads Time*  Min. : 1.00 Min. :17.08  1st Qu.: 3.00 1st Qu.:19.11  Median : 8.00 Median :23.91  Mean :18.14 Mean :30.51  3rd Qu.:24.00 3rd Qu.:37.22  Max. :64.00 Max. :59.91 | *N\_Threads Time*  Min. : 1.00 Min. : 2.453  1st Qu.: 3.00 1st Qu.: 4.255  Median : 8.00 Median : 8.289  Mean :18.14 Mean :17.737  3rd Qu.:24.00 3rd Qu.:22.643  Max. :64.00 Max. :59.617 |

Consideramos qué el número máximo de hilos es 64 debido que a partir de este número de hilos **restore2.c empieza a ser menos eficiente.**

**A picture containing logo

Description automatically generatedSpeedup**

El speedup indica la ganancia de velocidad que consigue el algoritmo paralelo con respecto a un algoritmo secuencial.

Siendo t(n) el mejor algoritmo secuencial conocido o el algoritmo paralelo ejecutado en 1 procesador

**A picture containing text

Description automatically generated**

**Eficiencia**

La eficiencia mide el grado de aprovechamiento que un algoritmo paralelo hace de un computador paralelo.

**restore1.c**

**restore2.c**

**Ejemplo de orden bash**

**Text

Description automatically generated**

Nota: Adjuntamos órdenes de trabajo (\*.sh) en el fichero.

# Ejercicio 6

**En este ejercicio hay que hacer que cada hilo muestre información sobre las iteraciones que le ha tocado procesar de un bucle paralelizado. En concreto, partimos de la versión paralela del ejercicio 2, donde, en cada iteración del bucle y, se reparten las iteraciones del bucle y2 entre los hilos. En principio habría que hacer que, en cada iteración del bucle y, cada hilo muestre un mensaje con su identificador, cuántas iteraciones ha procesado del bucle y2 (en esa iteración del bucle y) y cuáles han sido la menor y mayor distancias que ha encontrado en esas iteraciones. Sin embargo, para evitar que salgan demasiados mensajes por pantalla, haz que solo se muestren los mensajes correspondientes a la primera iteración del bucle y.**

//\*Completar/

# Anexos

### Estructura de archivos

* *sh-scripts* 
  + *restore0.sh*
  + *restore1.sh*
  + *restore2.sh*
  + *restore3.sh*
* *source-code* 
  + *c-source*
    - * *restore0.c*
      * *restore1.c*
      * *restore2.c*
      * *restore3.c*
  + *r-source*
    - *r-visualization.R*
  + *sh-creator*
    - *sh-creator.c*
    - *addF.h*
    - *demo.jpg*
    - *headerFile.h*
    - *sh-creator*
* *time-logs*
  + time-log-restore0.txt
  + time-log-restore1.txt
  + time-log-restore2.txt
  + time-log-restore3.txt

# 