TRABAJO GLOBAL INTEGRADOR

Fecha de Inicio: 17/10/2023 Fecha de entrega: 14/11/2023
Modalidad de entrega: a) Código fuente. b) Diseño, ejecución y análisis experimental. C) Informe final impreso y Coloquio.
Docentes a cargo: Dra. Paola Caymes Scutari / Dr. Germán Bianchini
Modalidad: Grupal / Presencial
Modalidad de evaluación: Evaluación continua y Evaluación final sumativa (entregas y coloquio. Valoración mediante Escalas de Calificación y Rúbrica). Puntaje total necesario para aprobar: 60pts.

***TEMAS*: Integración de los saberes adquiridos desde la Unidad 1 a hasta la Unidad 5.**
***OBJETIVOS*: Desarrollar en forma grupal una solución paralela para un problema dado incluyendo el ciclo de vida completo de diseño, implementación, validación y evaluación de rendimiento. Desarrollar habilidades de comunicación oral y escrita.**

En grupo, seleccionar uno de los temas propuestos (1, 2 o 3) y abordar los siguientes incisos:

a) (40 pts) Diseñar un algoritmo paralelo que resuelva el problema seleccionado. En base al algoritmo secuencial deben identificarse las posibilidades de paralelización, a partir de las cuales debe abordarse el diseño eligiendo las estrategias que resulten más adecuadas. La implementación debe contemplar un estilo de programación apropiado, y debe tener en cuenta la calidad y la eficiencia de la programación. El programa debe funcionar en el cluster para distintos parámetros de entrada.

b) (30 pts) Explicar las características de la solución de software realizada, a partir del diseño experimental y la ejecución de los experimentos. Incluir análisis de **speedup**, **eficiencia**, **balanceo de carga** y **escalabilidad**. El diseño experimental debe contemplar los aspectos necesarios para poder evaluar los índices mencionados.

c) (30 pts) Presentar el desarrollo de software en un coloquio final, acompañado de un informe, siguiendo el formato de artículo utilizado en el TP3, que condense todo el proceso de desarrollo: problema a resolver, solución secuencial, identificación de paralelismo, elección de estrategias y diseño de la solución paralela, implementación, diseño experimental, resultados, análisis de rendimiento, y conclusiones. Se valorará el nivel de organización, responsabilidad y colaboración en las tareas intragrupales.

TEMA 1: Diseñar un algoritmo paralelo que resuelva la multiplicación de matrices, con la forma C = A∗B, con A y B no necesariamente cuadradas.

Conceptualmente, la multiplicación de matrices funciona de la siguiente manera: Dadas una matriz A(MxR) de M filas y R columnas, donde cada uno de sus elementos se denota $\boldsymbol{a_{ij}}$ *con* $1 \leq \boldsymbol{i} \leq \boldsymbol{M}$, y $1 \leq \boldsymbol{j} \leq \boldsymbol{R}$; y una matriz B(*R*×*N*) *de R filas y N columnas, donde cada uno de sus elementos se denota* $\boldsymbol{b_{ij}}$ *con* $1 \leq \boldsymbol{i} \leq \boldsymbol{R}$, *y* $1 \leq \boldsymbol{j} \leq \boldsymbol{N}$; *la matriz C resultante de la operación de multiplicación de las matrices* **A y B, C = A × B**, *es tal que cada uno de sus elementos que se denota como* $\boldsymbol{c_{ij}}$ *con* $\boldsymbol{1 \leq i \leq M}$, y $\boldsymbol{1 \leq j \leq N}$, *y se calcula de la siguiente manera:*

$$c_{ij} = \sum_{k=0}^{R-1} a_{ik} b_{kj}$$

TEMA 2: Diseñar un algoritmo paralelo que resuelva el ordenamiento denominado **Merge Sort** para una lista de elementos. Cada elemento debe ser representado mediante una estructura con varios campos, uno de los cuales se considera campo "clave", es decir que será el campo a considerar a la hora de efectuar el ordenamiento. Desarrollar un programa que implemente dicho algoritmo paralelo.

Conceptualmente, el ordenamiento ***Merge Sort*** *funciona de la siguiente manera:*

1. *Si la longitud de la lista es 0 ó 1, entonces ya está ordenada. En otro caso:*
2. *Dividir la lista desordenada en dos sublistas de aproximadamente la mitad del tamaño.*
3. *Ordenar cada sublista recursivamente aplicando el Merge Sort.*
4. *Mezclar las dos sublistas en una sola lista ordenada.*

TEMA 3: Diseñar un algoritmo paralelo que resuelva el ordenamiento denominado **Quick Sort** para una lista de elementos. Cada elemento debe ser representado mediante una estructura con varios campos, uno de los cuales se considera campo "clave", es decir que será el campo a considerar a la hora de efectuar el ordenamiento. Desarrollar un programa que implemente dicho algoritmo paralelo.

Conceptualmente, el algoritmo ***Quick Sort*** *trabaja de la siguiente manera:*

1. *Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.*
2. *Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.*
3. *La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.*
4. *Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.*