Computación Paralela (MC-8836)

Instituto Tecnológico de Costa Rica
Escuela de Computación
I semestre, 2022
Tarea Programada 3

Instrucciones[1]

- Debe crear una cuenta en `https://gitlab.com` y crear un repositorio **privado** llamado `mc-8836` para las tareas del curso. Si no lo ha hecho ya, envíe un correo con su usuario de `GitLab` al profesor `esteban.meneses@acm.org` y al asistente `alexsaenz@estudiantec.cr`.
- Agregue a los usuarios `emeneses` (Esteban Meneses) y `sr.alexe` (Álex Sáenz) como colaborador de su repositorio GIT en `GitLab`. Los permisos de acceso de los usuarios `emeneses` y `sr.alexe` deben ser al menos los de un *Reporter*.
- Los fuentes para esta tarea están en el directorio `hw3` del repositorio GIT del profesor en `GitLab`:
  `git clone https://gitlab.com/emeneses/mc-8836.git`
- Las interfaces funcionales del código fuente provisto deben respetarse.
- El único mecanismo de entrega de esta asignación es por medio de su repositorio GIT en `GitLab`.
- Debe crear una carpeta `hw3` en su repositorio GIT. Ese directorio debe contener los siguientes archivos:
  • Archivo `shear.cpp` que contiene el código paralelo que implementa el algoritmo *shear sort*.
  • Archivo `report.pdf` con un reporte resumiendo los resultados del programa (debe ser un archivo en formato PDF).
- El código paralelo debe correr en la supercomputadora `Kabré` del Centro Nacional de Alta Tecnología (CeNAT). Para poder compilar programas Cilk Plus debe ejecutar el comando:
  `module load gcc/7.2.0`
  También puede copiar estos comandos en el archivo `~/.bashrc` y se ejecutarán automáticamente.
- Se recomienda utilizar trabajos interactivos para el desarrollo de la tarea:
  `salloc --partition=nu --time=02:00:00`
- Se debe utilizar trabajos en lote para recoger información de desempeño:
  `sbatch file.slurm`
- La fecha límite de entrega de esta asignación es el **viernes 25 de marzo a las 11:55pm**. Entregas extemporáneas **no** serán aceptadas.
- La presente asignación es estrictamente **individual**.

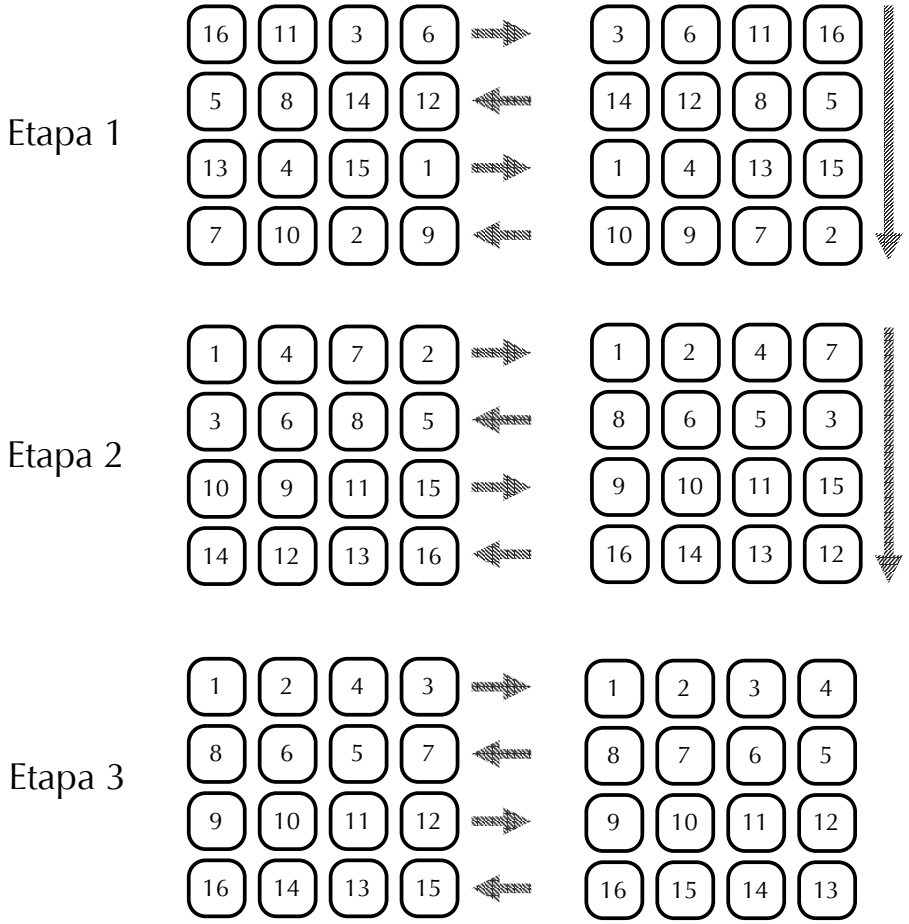
[1]Cada estudiante debe mostrar el estándar más alto de integridad académica. Presentar las ideas de otra persona sin el debido permiso es una clara violación de la política de integridad académica de la universidad y es una práctica que debilita las habilidades para resolver problemas en cualquier ámbito. Cualquier infracción a esta regla será llevada hasta sus últimas consecuencias.

Algoritmo Shear Sort

El algoritmo Shear Sort (también llamado Snake Sort) es un algoritmo paralelo diseñado originalmente para correr en una retícula bidimensional de procesadores. Sin embargo, el mismo algoritmo puede ser usado para ordenar N valores (donde N es un cuadrado perfecto) en una computadora con múltiples procesadores usando cualquier número de hilos de ejecución. El algoritmo Shear Sort ordena el arreglo original de $N = M^2$ elementos en una matrix cuadrada A de tamaño $M \times M$. El algoritmo procede a ejecutar $log_2(N)$ etapas. En cada etapa, las filas de la matrix son ordenadas (alternando ordenamiento creciente y decreciente) y luego las columnas son ordenadas (todas en orden creciente). El siguiente pseudoalgoritmo resume Shear Sort:

```
function shearSort(A, M):
    repeat log2(M*M) times:
        sortRowsAlternateDirection(A,M)
        sortColumns(A,M)
```

La matrix final A contiene los elementos ordenados si la matriz se recorre fila por fila en direcciones alternadas (empezando de izquierda a derecha). La figura de abajo presenta un ejemplo con las primeras 3 etapas de Shear Sort con una entrada de 16 elementos.



Escriba un programa paralelo con directivas de OpenMP que implemente el Shear Sort.
Asegúrese de seguir las siguientes instrucciones:

a) Debe implementar un algoritmo de ordenamiento para ordenar individualmente las filas y columnas (este algoritmo puede ser paralelo también).

b) Debe agregar directivas de OpenMP en el código con el fin de obtener la *mejor* forma de paralelizarlo.

c) Escriba su código usando el archivo provisto `shear_skeleton.cpp`, renombrándolo como `shear.cpp`.

d) Genere una versión secuencial de su código al compilar sin la opción `-fopenmp` y remover cualquier llamada a la biblioteca de funciones de OpenMP que pueda estar utilizando.

e) Su programa debe correr con los siguientes parámetros:
`./shear N`
or
`./shear N file`
El primer caso crea un arreglo aleatorio (almacenado como una matriz) de tamaño N, mientras que el segundo caso lee un arreglo almacenado en un archivo. En cualquier caso, N debe ser un cuadrado perfecto.

f) Use el siguiente comando para analizar el desempeño de su código usando un número diferente de hilos:
`OMP_NUM_THREADS=16 ./shear N`

Informe

Debe crear un informe con las siguientes secciones:

1. Una estrategia general del esfuerzo de paralelización. ¿Qué patrones de programación paralela usó en la solución. ¿Qué operaciones de OpenMP parecieron las más apropiadas para el programa?

2. Análisis de *speedup*. Presente un gráfico con el *speedup* de su solución paralela usando 2, 4, 8, 16, 32, 64, 128 y 256 *hilos*. Este gráfico debe tener el número de *hilos* en el eje `x` y el *speedup* en el eje `y`. Use el tiempo de ejecución del programa secuencial para calcular el *speedup*. Debe escoger un tamaño de problema *interesante* para su análisis.

3. Análisis de eficiencia. Presente un gráfico con la eficiencia de su solución paralela usando 2, 4, 8, 16, 32, 64, 128 y 256 *hilos*. Este gráfico debe tener el número de *hilos* en el eje `x` y la eficiencia en el eje `y`. Use el tiempo de ejecución del programa secuencial para calcular la eficiencia. Debe escoger un tamaño de problema *interesante* para su análisis.

Calificación

Se evaluará la correctitud, claridad y eficiencia del código escrito, así como el apego a las directrices establecidas. La división de la evaluación es la siguiente:

- Programación Shear Sort: 70 %
- Informe con resultados: 30 %