**Evaluación Experimental de Desempeño del Algoritmo K-Means Paralelizado con OpenMP**

**1. Introducción**

En este informe se presenta la evaluación experimental de un algoritmo **K-Means** implementado en C++ y paralelizado con **OpenMP**. El objetivo es **comparar** el rendimiento de la versión paralela frente a una versión serial, analizando cómo varía el **tiempo de ejecución** y el **speed-up** al modificar la cantidad de puntos en el dataset y el número de hilos disponibles.

**2. Definición del Experimento**

1. **Parámetros Principales**
   * **Número de puntos** (nPointsnPointsnPoints): {100000, 200000, 300000, 400000, 600000, 800000, 1000000}
   * **Número de hilos** (nThreadsnThreadsnThreads): {1, 4, 8, 16}
   * **Número de clusters** (kkk): 5 (fijo para todas las pruebas).
   * **Iteraciones del algoritmo**: se repite K-Means hasta que no haya cambios en la asignación de puntos o se alcance un máximo de 100 iteraciones.
2. **Repetición de Corridas**
   * Para cada combinación de (nPoints,nThreads)(nPoints, nThreads)(nPoints,nThreads), se realizaron **10 corridas** y se promediaron los tiempos de ejecución.
3. **Métrica de Desempeño**
   * **Tiempo de ejecución** (segundos): se midió tanto para la versión **serial** como para la **paralela**.
   * **Speed-up**: Speed-up=Tserial\Tparalelo
4. **Generación de Datos**
   * Los datos (puntos en 2D) se generaron usando una libreta (synthetic\_clusters.ipynb), produciendo archivos CSV con los puntos para cada valor de nPointsnPointsnPoints.

**3. Descripción del Equipo (Hardware y Software)**

* **Hardware**
  + **CPU**: Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz (2.21 GHz), con múltiples núcleos lógicos.
  + **Memoria RAM**: 16.0 GB (15.8 GB usable).
  + **Sistema Operativo**: Windows 11 Home.
* **Software**
  + **Compilador**: g++ con soporte OpenMP (-fopenmp).
  + **Scripts de automatización**: Python 3.x, con librerías pandas y matplotlib para recolectar y graficar datos.
  + **Código**: Un programa unificado que corre K-Means en modo serial y paralelo, midiendo tiempos.

**4. Resultados y Análisis**

La siguiente **tabla** muestra los resultados promediados de 10 corridas para cada configuración de (nPoints,nThreads)(nPoints, nThreads)(nPoints,nThreads). Se incluyen el **tiempo promedio** de la versión **serial** (avgTimeSerial), el de la versión **paralela** (avgTimeParallel) y el **speed-up** resultante.

| **nPoints** | **nThreads** | **avgTimeSerial (s)** | **avgTimeParallel (s)** | **Speed-up** |
| --- | --- | --- | --- | --- |
| 100000 | 1 | 0.097106 | 0.088541 | 1.097 |
| 100000 | 4 | 0.063171 | 0.029918 | 2.111 |
| 100000 | 8 | 0.072140 | 0.021162 | 3.409 |
| 100000 | 16 | 0.102966 | 0.025373 | 4.058 |
| 200000 | 1 | 0.188758 | 0.141984 | 1.329 |
| 200000 | 4 | 0.191433 | 0.112084 | 1.708 |
| 200000 | 8 | 0.147842 | 0.047716 | 3.098 |
| 200000 | 16 | 0.174518 | 0.029009 | 6.016 |
| 300000 | 1 | 0.229611 | 0.250300 | 0.917 |
| 300000 | 4 | 0.255771 | 0.109116 | 2.344 |
| 300000 | 8 | 0.193742 | 0.056179 | 3.449 |
| 300000 | 16 | 0.278580 | 0.055935 | 4.980 |
| 400000 | 1 | 0.303866 | 0.422456 | 0.719 |
| 400000 | 4 | 0.293521 | 0.151944 | 1.932 |
| 400000 | 8 | 0.351783 | 0.066189 | 5.315 |
| 400000 | 16 | 0.300975 | 0.067025 | 4.490 |
| 600000 | 1 | 0.657578 | 0.495545 | 1.327 |
| 600000 | 4 | 0.444667 | 0.255699 | 1.739 |
| 600000 | 8 | 0.559865 | 0.131509 | 4.257 |
| 600000 | 16 | 0.608762 | 0.093632 | 6.502 |
| 800000 | 1 | 0.655738 | 0.771372 | 0.850 |
| 800000 | 4 | 0.717847 | 0.367302 | 1.954 |
| 800000 | 8 | 0.791003 | 0.199400 | 3.967 |
| 800000 | 16 | 0.749311 | 0.137663 | 5.443 |
| 1000000 | 1 | 1.023774 | 1.109636 | 0.923 |
| 1000000 | 4 | 0.941896 | 0.422094 | 2.231 |
| 1000000 | 8 | 0.956904 | 0.249169 | 3.840 |
| 1000000 | 16 | 0.842347 | 0.225481 | 3.736 |

**Gráficas relevantes:**

**Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.**

**4.1 Análisis de Speed-up**

* En la mayoría de configuraciones con **4, 8 o 16 hilos**, se observa un **speed-up > 1**, lo que indica que la versión paralela es más rápida que la serial.
* Se registran algunos casos con speed-up < 1 (por ejemplo, 300000 puntos y 1 hilo, 400000 puntos y 1 hilo, 800000 puntos y 1 hilo). Esto puede deberse a la sobrecarga de la paralelización o a diferencias en la inicialización de los centroides.
* El **speed-up máximo** observado es de **6.502** para 600000 puntos y 16 hilos, seguido de **6.016** para 200000 puntos y 16 hilos.

**4.2 Tiempos Serial vs. Paralelo**

* A 1 hilo, la versión “paralela” a veces iguala o supera levemente el rendimiento de la serial (o lo empeora, dependiendo de la sobrecarga).
* A medida que aumentan los hilos (4, 8, 16), el tiempo de la versión paralela **disminuye** significativamente, al menos en datasets medianos/grandes.
* Se aprecian algunos altibajos debido a la aleatoriedad en la selección de centroides iniciales (que puede hacer que el algoritmo requiera más o menos iteraciones).

**4.3 Observaciones**

* El i7-10870H cuenta con 8 núcleos lógicos (o más si consideramos hyper-threading). Probar con 16 hilos puede saturar la capacidad de la CPU, pero en varios casos sigue obteniéndose un buen speed-up (ej. 600000 puntos y 16 hilos).
* Con **datasets pequeños** (p.ej., 100000 puntos), la sobrecarga de creación y sincronización de hilos puede afectar los resultados, aunque aún se ven speed-ups de 3–4 con 8 y 16 hilos.

**5. Conclusiones**

1. **Eficiencia de la Paralelización:**
   * El algoritmo K-Means paralelo muestra mejoras claras frente a la versión serial en la mayoría de las configuraciones, especialmente con **datasets más grandes** y al usar 8 o 16 hilos.
   * El speed-up puede llegar hasta valores de ~6.5, lo cual es un resultado muy favorable considerando la sobrecarga de sincronización.
2. **Casos con Speed-up < 1:**
   * Se observan cuando el número de puntos es relativamente grande o pequeño, pero con 1 hilo, la sobrecarga de OpenMP puede impactar negativamente.
   * También influyen la inicialización aleatoria de centroides y la variabilidad de iteraciones.
3. **Escalabilidad y Límite de Hilos:**
   * Aunque 16 hilos a veces brinda el mayor speed-up, en ciertos casos (por ejemplo, 400000 puntos) el speed-up baja ligeramente respecto a 8 hilos, debido a la sobrecarga adicional de sincronización.
4. **Trabajo Futuro:**
   * Se podrían probar **estrategias de inicialización** más sofisticadas (como K-Means++) para reducir la variabilidad.
   * Analizar la **distribución de iteraciones** en cada corrida.
   * Experimentar con **diferentes planificadores** de OpenMP (dynamic, guided) o con otras librerías de paralelismo (TBB, MPI).

**6. Archivo con Datos de los Experimentos**

Los datos completos de cada corrida promediada se encuentran en el archivo experiments\_results.csv, con las siguientes columnas:

* **nPoints**: número de puntos del dataset.
* **nThreads**: número de hilos usados en la versión paralela.
* **avgTimeSerial**: tiempo promedio (en segundos) de la versión serial.
* **avgTimeParallel**: tiempo promedio (en segundos) de la versión paralela.
* **speedUp**: cociente avgTimeSerial/avgTimeParalle

**En conclusión**, la versión paralela de K-Means logra mejoras sustanciales en tiempos de ejecución frente a la serial, especialmente con tamaños de datos grandes y al aprovechar el hardware multinúcleo del equipo de prueba.