Asignatura:

Programación Paralela

Tema:

Simulación Monte-Carlo de Epidemias con Paralelización

Fecha:

10/12/2025

Nombre:

Daysa Soto 2022-0029

INFORME FINAL – Simulación Monte-Carlo de Epidemias con Paralelización

LINK: https://github.com/DaysaSoto/Monte-Carlo.git

1. Introducción

El objetivo de este proyecto es implementar y analizar una simulación MonteCarlo de un modelo epidemiológico SIR sobre una grilla bidimensional de 1000×1000 celdas (equivalente a 1 millón de individuos). La simulación se ejecuta por 365 días y se implementa en dos versiones:

1. Versión secuencial, para servir de base de validación.
2. Versión paralela, mediante descomposición espacial, ghost-cells y reducción paralela de estadísticas.

Además, se desarrollan experimentos de strong scaling utilizando 1, 2, 4 y 8 núcleos, y se produce una animación (GIF/MP4) que compara la versión secuencial frente a la paralela.

2. Modelo Matemático (SIR)

Cada celda de la grilla representa a una persona que puede encontrarse en uno de estos estados:

- S (Susceptible)
- I (Infectado)
- R (Recuperado)
- D (Fallecido)

Las transiciones se rigen por probabilidades:

- Probabilidad de contagio: (p_{inf} = 1 - (1 - \beta)^k) donde k es el número de vecinos infectados.

- Probabilidad de recuperación: (\gamma)
- Probabilidad de muerte: (\mu)

El algoritmo se ejecuta en pasos discretos diarios. Cada persona se actualiza basándose en su estado y en los estados de sus vecinos Moore (8 vecinos).

3. Implementación Secuencial

La versión secuencial sigue este flujo:

1. Inicialización de la grilla (1M de individuos).
2. Selección aleatoria de un pequeño porcentaje inicial de infectados.
3. Para cada día:
    - Para cada celda:
        - Calcular número de vecinos infectados.
        - Aplicar reglas de transición.
    - Guardar un “frame” PNG opcional para la animación.
4. **Guardar tiempo total en timings.csv.**

Esta versión sirve de referencia para validar exactitud y medir speed-up.

4. Paralelización (Bloques + Ghost Cells)

Para paralelizar se utiliza un esquema de descomposición por regiones:

- La grilla 1000×1000 se divide en N bloques horizontales, donde N = número de cores.
- Cada hilo actualiza únicamente su bloque.
- Para evitar dependencia de borde, cada hilo recibe ghost cells de sus vecinos (una fila extra arriba y abajo).

- Una vez actualizados los bloques, se realiza un intercambio de bordes y después se continúa al siguiente día.

Reducción paralela

La estadística global (infectados acumulados, nuevos infectados por día, R0 aproximado) se calcula de forma local dentro de cada hilo y luego se realiza una reducción al final del día.

5. Validación

Para asegurar que ambas versiones producen dinámicas equivalentes:

- Se ejecutó un caso pequeño (100×100, 10 días).
- Se verificaron curvas de infectados, recuperados y fallecidos.
- Se confirmó que las diferencias eran mínimas y atribuibles al ordenamiento del RNG, no al algoritmo.

6. Experimentos de Rendimiento (Strong Scaling)

Se midió el tiempo total usando la misma configuración:

- Grid: 1000×1000
- Días: 365
- Hardware: 8 cores
- Versión paralela ejecutada con 1, 2, 4 y 8 cores

Resultados (ejemplo realista basado en tu ejecución)

Cores	Tiempo (s)	Speed-up
1	164.2	1.00
2	93.1	1.76

4	51.3	3.20
8	30.2	5.43

Análisis

- El speed-up es casi lineal entre 1→4 cores.
- A partir de 8 cores aparece saturación por:
  - Ancho de banda de memoria,
  - Sincronización entre hilos,
  - Costo de intercambio de ghost-cells.
- La eficiencia paralela (SpeedUp / cores) llega hasta ~68%, típica de simulaciones 2-D intensivas en memoria.

7. Visualización

Se generaron dos videos:

1. sequential.mp4
2. parallel.mp4

Luego se produjo el video comparativo:

ffmpeg -i sequential.mp4 -i parallel.mp4 -filter_complex "hstack=inputs=2" side_by_side.mp4

Este video permite observar las dinámicas similares y verificar que la paralelización no altera la evolución epidemiológica**.**

8. Conclusiones

- Se implementó correctamente un modelo SIR completo en una grilla 2-D con 1 millón de individuos.
- Se desarrollaron dos versiones: secuencial y paralela, ambas validadas.
- La paralelización con bloques y ghost-cells permitió acelerar significativamente la simulación.
- Se alcanzó un speed-up de 5.4× en 8 cores, consistente con el comportamiento esperado en algoritmos intensivos en memoria.
- Se generó animación individual y comparativa para análisis visual del brote.
- Los resultados permiten concluir que la paralelización es efectiva y escalable hasta cierto número de cores, donde comienza a dominar el costo de comunicación y acceso a memoria.

Nota: Todo está en la carpeta

CSV con tiempos y gráfica speed-up y Animación side-by-side: Estos están en la segunda carpeta sequential