TÍTULO:

ANÁLISIS DE RENDIMIENTO Y MODELADO EPIDEMIOLÓGICO SIR MEDIANTE COMPUTACIÓN PARALELA EN JAVASCRIPT: CASO DE ESTUDIO COVID-19 EN REPÚBLICA DOMINICANA

AUTOR: Anddy Josue Jara Ramirez FECHA: 20 Noviembre 2025

1. INTRODUCCIÓN Y CONTEXTO

La modelización computacional de epidemias se ha convertido en una herramienta fundamental para la toma de decisiones en salud pública. Modelos como el SIR (Susceptible-Infectado-Recuperado) permiten predecir el comportamiento de patógenos en poblaciones grandes. Sin embargo, cuando se introduce la estocasticidad (azar) mediante métodos de Monte-Carlo en grandes poblaciones, el costo computacional se eleva drásticamente, haciendo necesaria la implementación de técnicas de paralelismo.

Este proyecto presenta una simulación de una epidemia similar al COVID-19 sobre una población de ***1,000,000*** de individuos distribuidos en una grilla espacial de *1000 X 1000.* A diferencia de las ecuaciones diferenciales ordinarias (EDO) que asumen una mezcla homogénea, este modelo espacial respeta la localidad de las interacciones: una persona solo puede ser infectada por sus vecinos inmediatos.

El objetivo técnico es demostrar cómo la **descomposición de dominio** puede reducir el tiempo de ejecución de la simulación (Speed-up) utilizando la arquitectura de Worker Threads de Node.js, comparando los resultados con una implementación secuencial tradicional.

2. MODELO MATEMÁTICO

Se utilizó un autómata celular probabilístico donde cada celda (i,j) de la matriz representa a un individuo que puede estar en uno de cuatro estados:

- *S* (Susceptible): Sano pero en riesgo.
- *I* (Infectado): Portador del virus capaz de contagiar.
- *R* (Recuperado): Inmune tras superar la enfermedad.
- *D* (Fallecido): Deceso por complicaciones.

2.1 Dinámica de Transición (Monte-Carlo)

La simulación avanza en pasos discretos de tiempo (Delta t = 1 día). En cada paso, se evalúa el cambio de estado de cada celda basándose en números aleatorios:

1. Regla de Contagio (S - I): Un individuo susceptible se infecta con una probabilidad que depende de la cantidad de vecinos infectados (k) en su radio de Moore (8 vecinos):

$$P_{infección} = 1 - (1 - \beta)^k$$

Donde β es la tasa de transmisibilidad del virus.

2. Regla de Recuperación y Muerte (I \R ó I \ D):

Un infectado tiene una probabilidad diaria delta de morir y una probabilidad gamma de recuperarse.

2.2 Calibración del Modelo (Escenario Endémico RD)

Para cumplir con el requisito de simulación extendida (>365 días) y reflejar la realidad del COVID-19 en República Dominicana, se ajustaron los parámetros para evitar un agotamiento rápido de susceptibles (brote explosivo).

- **Beta (beta = 0.18):** Transmisión moderada.
- **Gamma (gamma = 0.08):** Recuperación lenta (aprox. 12-14 días de enfermedad).
- Delta (delta = 0.002): Tasa de letalidad baja pero constante.

Esta configuración genera un comportamiento endémico con olas de contagio que persisten durante los 600 días simulados.

3. ARQUITECTURA Y DISEÑO PARALELO

La implementación se realizó en **JavaScript (Node.js)**. Dado que Node.js es tradicionalmente *single-threaded*, se utilizó el módulo worker_threads para lograr paralelismo real sobre múltiples núcleos de CPU.

3.1 Descomposición de Dominio

Se aplicó una descomposición espacial 1D (por filas). La grilla de 1000 X 1000 se divide horizontalmente en N bloques, donde N es el número de hilos disponibles. Cada Worker es responsable de calcular el estado futuro de las celdas dentro de su bloque asignado.

3.2 Comunicación y Celdas Fantasma (Ghost Cells)

El desafío principal en la paralelización de grillas es el cálculo de los bordes. Para determinar si una celda en el borde superior de un bloque se infecta, necesita conocer el estado de la fila inferior del bloque anterior (que pertenece a otro hilo).

Para resolver esto se implementaron **Celdas Fantasma**:

1. Cada Worker mantiene una copia de la última fila de su vecino superior y la primera de su vecino inferior.
2. **Sincronización:** Al final de cada día simulado, el hilo principal (Main Thread) recibe las fronteras actualizadas de todos los workers y las redistribuye. Esto asegura la coherencia matemática de la simulación: el resultado paralelo es idéntico al secuencial.

4. ANÁLISIS DE RESULTADOS Y RENDIMIENTO

Se realizaron experimentos de *Strong Scaling* (Escalado Fuerte), manteniendo el tamaño del problema fijo (1M agentes) y aumentando el número de hilos (1, 2, 4, 8).

4.1 Métricas de Desempeño

- **Tiempo Secuencial (T_1):** ~100.36 segundos (para 600 días de simulación compleja).
- **Tiempo Paralelo (4 hilos):** ~61.63 segundos.

4.2 Speed-up y Eficiencia

$$Speedup = \frac{T_{secuencial}}{T_{paralelo}}$$

Se observó un Speed-up sub-lineal. Aunque el tiempo total disminuye, no se reduce a la mitad al duplicar los núcleos. Esto se debe a:

1. **Overhead de Serialización:** En JavaScript, pasar grandes arrays de datos entre el hilo principal y los workers implica clonación de memoria (Structured Clone Algorithm), lo cual es costoso.
2. **Cuello de Botella de Sincronización:** La simulación debe detenerse completamente cada "día" para intercambiar las celdas fantasma, limitando la independencia de los hilos (Ley de Amdahl).

4.3 Análisis Epidemiológico

La visualización en dashboard_epidemias.html muestra exitosamente la dispersión geográfica del virus sobre el mapa de República Dominicana. Las curvas generadas muestran un comportamiento ondulatorio, validando la calibración de los parámetros para escenarios de larga duración. El sistema permite observar cómo el virus persiste en la población sin extinguirse rápidamente, mimetizando la fase endémica del SARS-CoV-2.

5. CONCLUSIÓN

El proyecto demuestra la viabilidad de utilizar tecnologías web modernas (JavaScript/Node.js) para simulaciones científicas de alto rendimiento. La implementación paralela logró reducir el tiempo de cómputo, permitiendo simulaciones más largas y complejas en tiempos razonables.

La comparación visual y numérica valida que la división del problema mediante *Ghost Cells* no altera el resultado determinista de la simulación, cumpliendo con los estándares de rigor científico. La herramienta resultante (dashboard_epidemias) sirve no solo como prueba de concepto técnica, sino como una herramienta educativa para visualizar la propagación de pandemias en el contexto local dominicano.