Implementación del Juego de la Vida de Conway

Autómata Celular con Representación de Bits y Paralelización

Integrantes:
Samuel Andrés Ariza Gómez
Andrés Vélez Rendón
Juan Pablo Mejía Pérez

1 de junio de 2025

Índice

1. Introducción

Este documento presenta una implementación del Juego de la Vida de Conway desarrollada en lenguaje C. El autómata celular opera sobre un tablero de 64×64 células durante 100 generaciones, aplicando las reglas estándar de supervivencia, muerte y nacimiento.

La implementación utiliza tipos de datos `uint64_t` para representar cada fila del tablero, donde cada bit corresponde al estado de una célula. Se emplea OpenMP para distribuir el procesamiento entre múltiples hilos de ejecución.

2. Arquitectura del Sistema

2.1. Representación de Datos

El tablero se define mediante la siguiente estructura:

```
1 typedef uint64_t BoardRow;
2 typedef struct {
3     BoardRow grid[BOARD_ROWS];
4 } GameBoard;
```

Cada fila del tablero se almacena en un entero de 64 bits, donde el bit en la posición `i` indica el estado de la célula en la columna `i` (1 = viva, 0 = muerta).

2.2. Organización del Código

El proyecto se estructura en los siguientes archivos:

- **src/**: Contiene los archivos de implementación
  - `bitboard.c`: Definición de la estructura `GameBoard`
  - `game.c`: Funciones de inicialización y cálculo de generaciones
  - `util.c`: Funciones de visualización y control
  - `main.c`: Función principal y coordinación
- **include/**: Contiene los archivos de cabecera
  - `bitboard.h`: Declaraciones para el manejo del tablero
  - `game.h`: Declaraciones de funciones de simulación
  - `util.h`: Declaraciones de funciones utilitarias
  - `config.h`: Definición de constantes y parámetros

3. Implementación

3.1. Inicialización del Tablero

La función de inicialización genera un tablero aleatorio distribuyendo el trabajo entre hilos:

```c
void initialize_random_board(GameBoard *board) {
    srand((unsigned)time(NULL));
    #pragma omp parallel for num_threads(NUM_THREADS)
    for (int row_idx = 0; row_idx < BOARD_ROWS; row_idx++) {
        BoardRow current_row = 0;
        for (int col_idx = 0; col_idx < BOARD_COLS; col_idx++) {
            if (rand() % 2) {
                current_row |= (1ULL << col_idx);
            }
        }
        board->grid[row_idx] = current_row;
    }
}
```

3.2. Conteo de Vecinos

El algoritmo cuenta los vecinos vivos de cada célula considerando la topología toroidal:

```c
int count_living_neighbors(const GameBoard *board, int row, int
   col) {
    int count = 0;
    int fila_anterior = (row - 1 + BOARD_ROWS) % BOARD_ROWS;
    int fila_siguiente = (row + 1) % BOARD_ROWS;
    int columna_anterior = (col - 1 + BOARD_COLS) % BOARD_COLS;
    int columna_siguiente = (col + 1) % BOARD_COLS;

    count += (board->grid[fila_anterior] >> columna_anterior) &
       1;
    count += (board->grid[fila_anterior] >> col) & 1;
    count += (board->grid[fila_anterior] >> columna_siguiente) &
       1;
    count += (board->grid[row] >> columna_anterior) & 1;
    count += (board->grid[row] >> columna_siguiente) & 1;
    count += (board->grid[fila_siguiente] >> columna_anterior) &
       1;
    count += (board->grid[fila_siguiente] >> col) & 1;
    count += (board->grid[fila_siguiente] >> columna_siguiente) &
        1;

    return count;
}
```

3.3. Cálculo de la Siguiente Generación

La función aplica las reglas del Juego de la Vida a cada célula:

```c
void compute_next_generation(const GameBoard *current, GameBoard
   *next) {
    #pragma omp parallel for num_threads(NUM_THREADS)
    for (int row_idx = 0; row_idx < BOARD_ROWS; row_idx++) {
```

```
        BoardRow next_gen_row = 0;
        for (int col_idx = 0; col_idx < BOARD_COLS; col_idx++) {
            int cantidad_vecinos_vivos = count_living_neighbors(
                current, row_idx, col_idx);
            int esta_viva = (current->grid[row_idx] >> col_idx) &
                1;

            if (esta_viva && (cantidad_vecinos_vivos == 2 ||
                cantidad_vecinos_vivos == 3)) {
                next_gen_row |= (1ULL << col_idx);
            } else if (!esta_viva && cantidad_vecinos_vivos == 3)
                {
                next_gen_row |= (1ULL << col_idx);
            }
        }
        next->grid[row_idx] = next_gen_row;
    }
}
```

3.4. Función Principal

El programa principal controla el flujo de ejecución:

```
int main(int argc, char *argv[]) {
    GameBoard game_board;
    int is_manual_mode = (argc > 1 && strcmp(argv[1], "manual")
       == 0);

    initialize_random_board(&game_board);
    run_simulation(&game_board, GENERATIONS, is_manual_mode);

    return 0;
}
```

4. Características del Sistema

4.1. Uso de Memoria

La representación mediante `uint64_t` utiliza 512 bytes para almacenar el tablero de 64×64 células (64 enteros × 8 bytes cada uno), comparado con 4096 bytes que requeriría un arreglo de `char`.

4.2. Paralelización

El programa utiliza OpenMP con la directiva `#pragma omp parallel for` para distribuir el procesamiento de filas entre múltiples hilos. El número de hilos se define en la constante `NUM_THREADS`.

4.3. Topología

El tablero implementa topología toroidal, donde los bordes se conectan entre sí. Esto se logra mediante el uso del operador módulo (`%`) en el cálculo de índices de vecinos.

5. Compilación y Uso

5.1. Requisitos

- Compilador C compatible con OpenMP
- Bibliotecas estándar de C
- Sistema que soporte bibliotecas POSIX (`unistd.h`)

5.2. Compilación

```
make
```

5.3. Ejecución

El programa acepta dos modos de operación:

```
./game_of_life          # Modo autom tico
./game_of_life manual   # Modo manual
```

En modo automático, las generaciones se muestran con intervalos de tiempo. En modo manual, el usuario debe presionar Enter para avanzar a la siguiente generación.

6. Configuración

Los parámetros del sistema se definen en `config.h`:

- `BOARD_ROWS`: Número de filas (64)
- `BOARD_COLS`: Número de columnas (64)
- `GENERATIONS`: Número de generaciones a simular (100)
- `NUM_THREADS`: Número de hilos para OpenMP

7. Reglas del Autómata

El programa implementa las reglas estándar del Juego de la Vida:

- Una célula viva con 2 o 3 vecinos vivos permanece viva
- Una célula viva con menos de 2 o más de 3 vecinos muere
- Una célula muerta con exactamente 3 vecinos vivos nace

8. Repositorio

El código fuente de este proyecto está disponible en GitHub:
https://github.com/AndresVelezR/game_of_life.git

9. Referencias

Referencias

[1] Conway, J. H. (1970). The Game of Life. *Scientific American*, **223**(4), 120-123.

[2] OpenMP Architecture Review Board. (Latest version). *OpenMP Application Programming Interface*. URL del sitio web de OpenMP (e.g., https://www.openmp.org/specifications/)