Trabajo práctico 1 Sistemas Operativos 1

Leonid Chanco Castillo

Juego de la vida.

El Juego de la Vida fue planteado por el matemático John Horton Congway en 1970. Es un juego sin jugadores que planteado como un autómata celular. Esto significa que la única interacción está dada al comienzo del juego, planteando un tablero inicial, y que luego el sistema evolucionará siguiendo las reglas establecidas.

Reglas

El juego presenta un tablero bidimensional infinito compuesto por células. Las células pueden estar vivas o muertas. Al ser un tablero bidimensional e infinito, cada célula tiene 8 células vecinas.
En cada momento, las células pasan a estar vivas o muertas siguiendo las reglas:

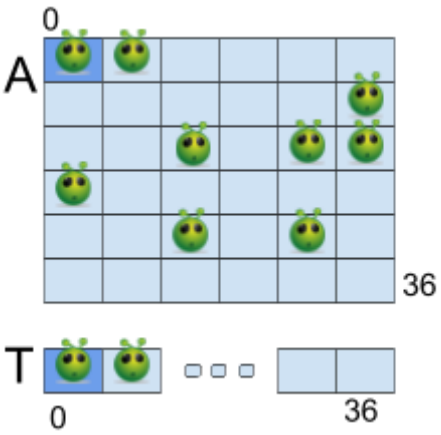
-Toda célula viva con 2 o 3 vecinos vivos sobrevive.
-Toda célula muerta con exactamente con 3 vecinos vivo revive.
- El resto de las células vivas mueren en la siguiente generación, como a su vez

el resto de las células muertas se mantienen muertas.

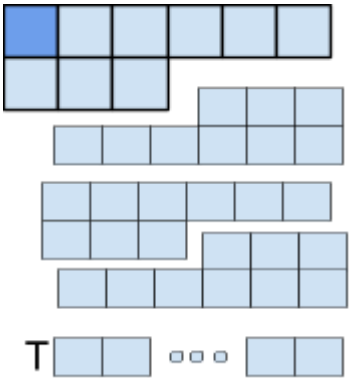
El patrón inicial del tablero se le suele llamar semilla. La primer generación es el resultado de aplicar las 3 reglas antes descritas a todas las células de la semilla, las transiciones se dan de forma simultánea. Las reglas se siguen aplicando de la misma manera para obtener futuras generaciones.

Implementación.

Para dicha implementación primero habíamos pensado en paralelizar cada célula del tablero del juego, es decir, crear un hilo por cada célula puesto que estos son independiente y así evitaremos tener deadlocks. Pero el problema que esto conlleva es que a un tablero de tamaño muy grande necesitamos crear la cantidad de (row * col) hilos y nuestro cpu dispone de una cantidad limitada de hilos, es decir, tenemos una cota superior de la cantidad de hilos que podemos crear.
Independientemente de esto también la creación de hilos es costosa y crear muchos hilos a veces no es la solución porque se introduce más procesamiento en la creación de hilos y sincronización que en el cómputo en sí.
Finalmente optamos por dividir el tablero por la cantidad de unidades de procesamientos disponibles que tenemos , es decir, lanzamos CANT_HILOS , donde CANT_HILOS es igual a la cantidad de cores de nuestro CPU. Asi asignándole una porción de células para cada hilo, consecuentemente ninguno de los hilos está esperando la liberación de algún recurso ya que cada uno tiene asignado una partición independiente del conjunto de células, por lo tanto no se cumple la cuarta condición para estar en un deadlock.

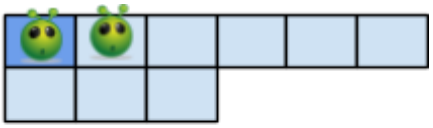


Representamos el tablero de la células como un arreglo **A** de 0 a 36 y Creamos otro arreglo temporal **T** de 0 a 36 .

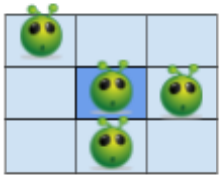


Particionamos el arreglo **A** en cantidad de nucleos del CPU *ejemplo* ***4***.

Lanzamos 1 hilo por cada partición

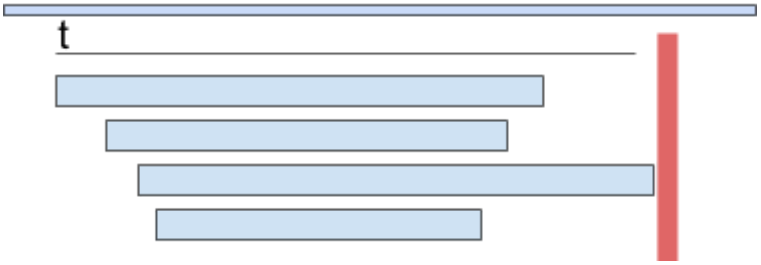


Por cada hilo, **en paralelo** recorremos toda la partición única que le corresponde.



Por cada casillero visitado calculamos la cantidad de vecinos vivos *ejemplo 3 vivos para el de la imagen*.

Por lo tanto en el tablero **T** asignamos una célula viva. Así por cada índice del arreglo **A**



Consideramos **t** como el punto donde se comienzan a lanzar los hilos, estos independientemente del tiempo que demoren en ejecutarse esperan en la línea roja "pthread_join" a que todos los hilos terminen, asi sincronizandose y dando lugar a que todo el arreglo temporal **T** tenga la información de la nueva generación del tablero.

Luego se procede a reemplazar el tablero **A** por el tablero **T**, dando lugar a un nuevo ciclo.