INF-384: Taller de Modelos y Métodos Cuantitativos
Taller 2
Lobos y Conejos

Prof. Álvaro Salinas

5 de Noviembre de 2020

1. Descripción y Marco Teórico

En el siguiente taller, usted realizará un análisis del paralelismo ofrecido por CUDA al resolver un determinado problema. Para lograrlo, deberá implementar CUDA kernels que permitan ejecutar un procesamiento paralelo sobre un conjunto de datos, para posteriormente comparar su desempeño con una versión secuencial que se ejecute en la CPU. Los objetivos principales de esta experiencia consisten en evaluar sus conocimientos sobre el manejo de hebras e índices, reducción en memoria compartida y operaciones atómicas.

Lobos y Conejos

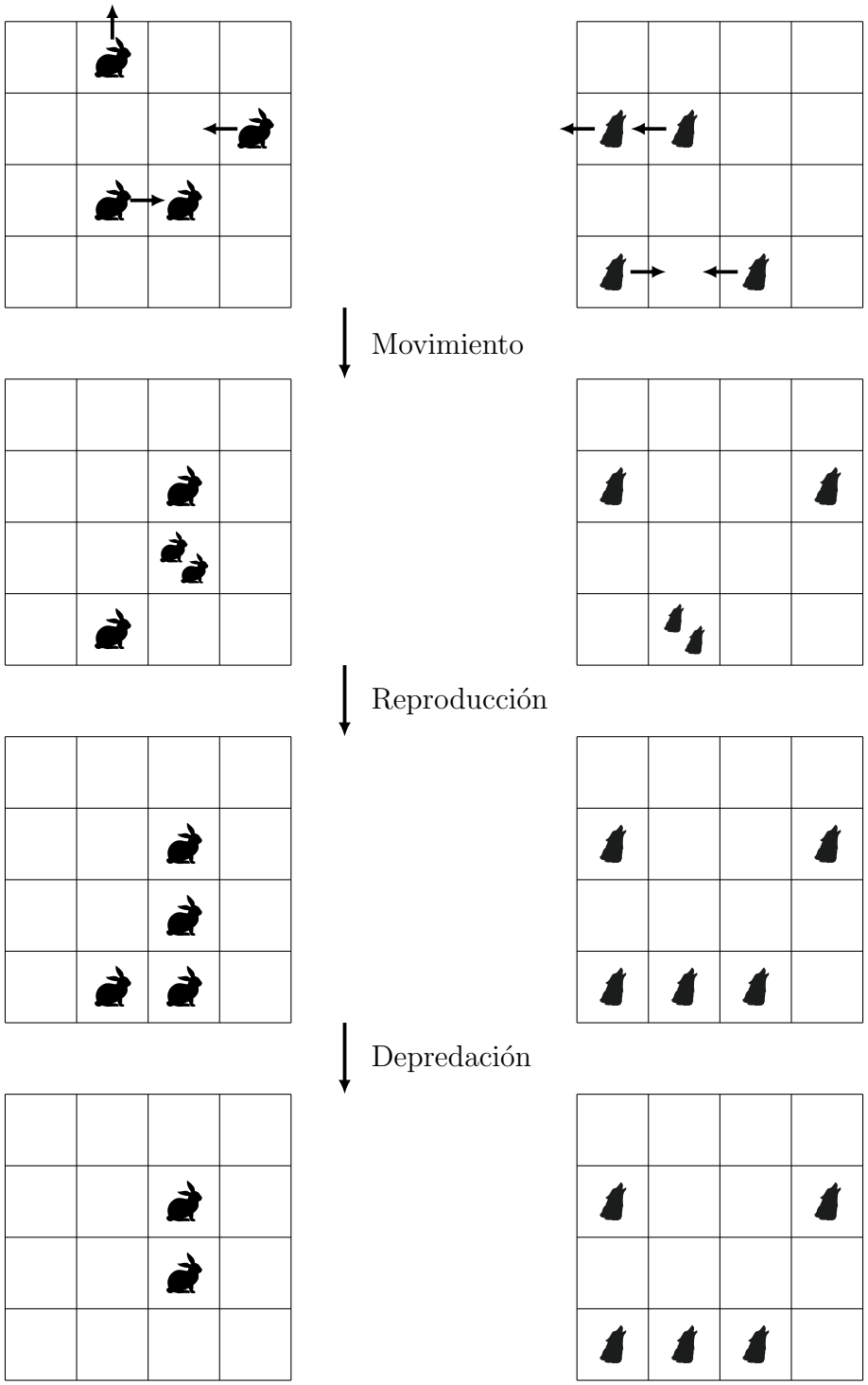
En el presente taller, vamos a simular un escenario donde conviven lobos y conejos siguiendo algunas reglas específicas. Este "juego" consiste en un tablero dividido en casillas uniformes representadas por dos arreglos. En el primero de ellos encontraremos las posiciones de los conejos, mientras que en el segundo estarán almacenadas las ubicaciones de los lobos.

La simulación consistirá en rondas donde se aplicarán reglas específicas:

- Al inicio de cada ronda se realizará el movimiento. Cada lobo y conejo generará un número entero aleatorio de 0 a 4, el cual determinará su acción. Respectivamente, el conejo podrá quedarse en la misma casilla (0) o moverse hacia la derecha (1), hacia arriba (2), hacia la iziquierda (3) o hacia abajo (4). Considere condiciones de borde periódicas, es decir, si por ejemplo un lobo en el borde inferior intenta moverse hacia abajo, aparecerá en la misma columna por el borde superior.
- Al finalizar la etapa anterior, se procede a la etapa de reproducción:
  - Si dos conejos coinciden en la misma casilla, y no hay conejos en las casillas superior e inferior, entonces serán reemplazados por tres conejos en dichas casillas.
  - Si dos lobos coinciden en la misma casilla, y no hay lobos en las casillas a la izquierda y derecha, entonces serán reemplazados por tres lobos en dichas casillas.
  - En ambos casos, en caso de ya haber animales en las casillas necesarias de forma previa, simplemente se agregarán nuevos en aquellas que estén vacías. Cualquiera sea el caso, la casilla central con los dos animales, solo contendrá uno de ellos al finalizar esta etapa.
- Al finalizar cada ronda, se debe simular el rol de depredador que poseen los lobos. Para realizar esto, deben desaparecer todos los conejos que se encuentren en la misma casilla que un lobo.

Al finalizar todas las rondas, el juego termina al contar la cantidad de lobos y conejos vivos.

Consideremos el siguiente ejemplo. El arreglo de conejos se encuentra a la izquierda y el de lobos a la derecha. Las flechas representan el movimiento determinado por el número aleatorio (si no hay flecha, el número es 0).



2. Archivos de Entrada

Los archivos que su programa recibirá tienen el siguiente formato:

- La primera línea contiene dos enteros M y N, correspondientes al número de filas y columnas del tablero respectivamente.
- Luego de la primera línea, existen M líneas con N valores 0, 1 o 2. El valor 0 indica que dicha casilla se encuentra vacía, mientras que los valores 1 o 2 indican la presencia de un conejo o lobo respectivamente. Esto significa que no existen casillas con más de un animal en el estado inicial.

3. Desarrollo

1. Implemente dos funciones secuenciales de C/C++: una que simule una ronda del juego y otra que cuente la cantidad de lobos y conejos vivos a partir de los arreglos. Simule 1000 rondas del juego llamando a la primera función dentro de un ciclo y mida el tiempo que demora este proceso. Al finalizar, mida el tiempo que le toma a la segunda función contar el número de lobos y conejos vivos.

2. Implemente uno o más CUDA kernels que simulen una ronda del juego. Al igual que lo realizado en el punto anterior, mida el tiempo que demora simular 1000 rondas del juego.

   [Pregunta] ¿Cuál de las dos implementaciones presenta un mejor desempeño? Comente al respecto.

   [Pregunta] ¿Enfrentó problemas relacionados a posibles condiciones de carrera? De ser así, descríbalos e indiqué como los solucionó.

3. Implemente un CUDA kernel que cuente la cantidad de lobos y conejos vivos mediante una reducción con memoria compartida. Mida el tiempo de su ejecución.

4. Implemente un CUDA kernel que cuente la cantidad de lobos y conejos vivos mediante el uso de operaciones atómicas. Mida el tiempo de su ejecución.

   [Pregunta] ¿Cuál es la versión más eficientre entre la secuencial y los dos kernels? Concluya respecto a los tiempos observados.

   [Informe] Presente sus resultados (número de conejos y lobos vivos).

   [Informe] Presente una tabla con los tiempos comparados tanto para la simulación de las 1000 rondas como para el conteo final de animales vivos.

4. Reglas y Consideraciones

Entrega

- Se debe realizar una entrega semanal del código al finalizar cada clase. Junto al código, se debe adjuntar un documento en donde se comente brevemente lo que se realizó en la respectiva sesión.
- Al finalizar el taller, se debe realizar una entrega final. Esta corresponde al código terminado más un informe en formato pdf. Se le ruega entregar un código ordenado.
- El informe debe contener:
  - Título y número del taller.
  - Nombre y rol de todos los integrantes del grupo.
  - Modelo y compute capability de la tarjeta gráfica que fue utilizada para ejecutar el código. Si se probó con más de una tarjeta gráfica, incluya los datos de todas y específique qué tarjeta se utilizó en cada pregunta y resultado reportado.
  - Desarrollo. Preocúpese de incluir cada ítem señalado con los tag [**Pregunta**] e [**Informe**] en el enunciado.
  - Conclusiones. Incluya aprendizajes, desafíos, comentarios, observaciones o supuestos que surgieron durante el desarrollo del taller.
- Cada entrega debe realizarse en un archivo de nombre Taller2-X.tar.gz (formatos rar y zip también son aceptados), donde X debe ser reemplazado por el número de su grupo. Diríjase al documento Grupos publicado en el aula para consultar su número.
- En caso de copia, los grupos involucrados serán evaluados con nota 0. No hay problema en generar discusión y compartir ideas de implementación con sus compañeros, pero códigos copiados y pegados no serán aceptados.
- El no cumplimiento de estas reglas implica descuentos en su evaluación.

Consideraciones

- Para mediciones de tiempo, `clock()` de la librería time.h en caso de mediciones en CPU y `cudaEvent()` o Nvidia Visual Profiler para códigos de GPU. Trabaje con milisegundos $[ms]$.
- Utilice 256 hebras por bloque en su implementacion.
- Para la generación de números aleatorios, utilice la librería cuRAND.