INF-351: Computación de Alto Desempeño
Laboratorio 2
LGA en GPU

Prof. Álvaro Salinas

17 de Mayo de 2020

1. Descripción y Marco Teórico

En el siguiente laboratorio serán evaluados sus conocimientos sobre divergencia, uso de memoria global y accesos de memoria coalescentes.

Lattice Gas Automata

Lattice Gas Automata (LGA) es un tipo de autómata celular utilizado para realizar simulaciones de fluidos a través de la resolución de las conocidas ecuaciones de Navier-Stokes. Este método es el predecesor del actual ampliamente utilizado método de lattice Boltzmann. Si bien la teoría de dinámica de fluidos relacionada a este método no será revisada debido a que no es relevante para el desarrollo de este laboratorio, sí es necesario que analicemos su mecánica para comprender su implementación.

En esta ocasión, implementaremos un tipo especial de LGA, denominado modelo HPP. Éste consiste en una malla uniforme equiespaciada conformada por nodos que se encuentran a una distancia de 1 lattice unit (lu). En cada nodo es posible tener hasta 4 partículas, cada una asociada a una dirección. Dichas particulas solo pueden existir en un nodo y son representadas por un valor binario $f_i(\mathbf{x}, t)$ que representa la presencia (1) o ausencia (0) de una partícula en la i-ésima dirección del nodo en la posición $\mathbf{x} = (x, y)$ en el tiempo t. La Figura 1 presenta el esquema recién descrito.

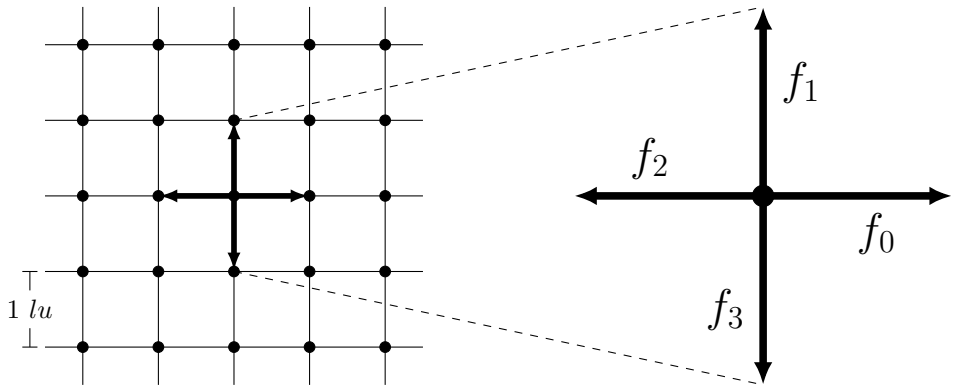

Figura 1: Esquema del modelo HPP.

El tiempo se discretiza en time steps (ts), i.e. $\Delta t = 1\ ts$ con $t_i = t_{i-1} + \Delta t$. Un time step del modelo HPP consiste en dos pasos denominados collision y streaming. El primero de ellos consiste en la representación de las posibles colisiones que pueden generarse entre partículas que llegan a un mismo punto desde direcciones opuestas, mientras que el paso de streaming modela el movimiento de las partículas desde un nodo a su vecino según la dirección de cada una de ellas.

Es importante notar que el paso de collision es un proceso local de cada nodo, ya que solo supone un reordenamiento de las partículas respecto a sus direcciones. Por otro lado, el paso de streaming no es un proceso local, pues existe interacción entre el nodo y su vecindad al trasladarse las partículas. En este contexto, solo son considerados dos casos posibles para el paso de collision, los cuales corresponden a la presencia de únicamente dos partículas en dirección opuesta, en cuyo caso deben ser rotadas en 90°. La Figura 2 contiene ejemplos ilustrativos de evolución en un time step (t_i a la izquierda y t_{i+1} a la derecha). Notar que el segundo y tercer ejemplo presentados en la figura son los únicos casos en los que hay collision.

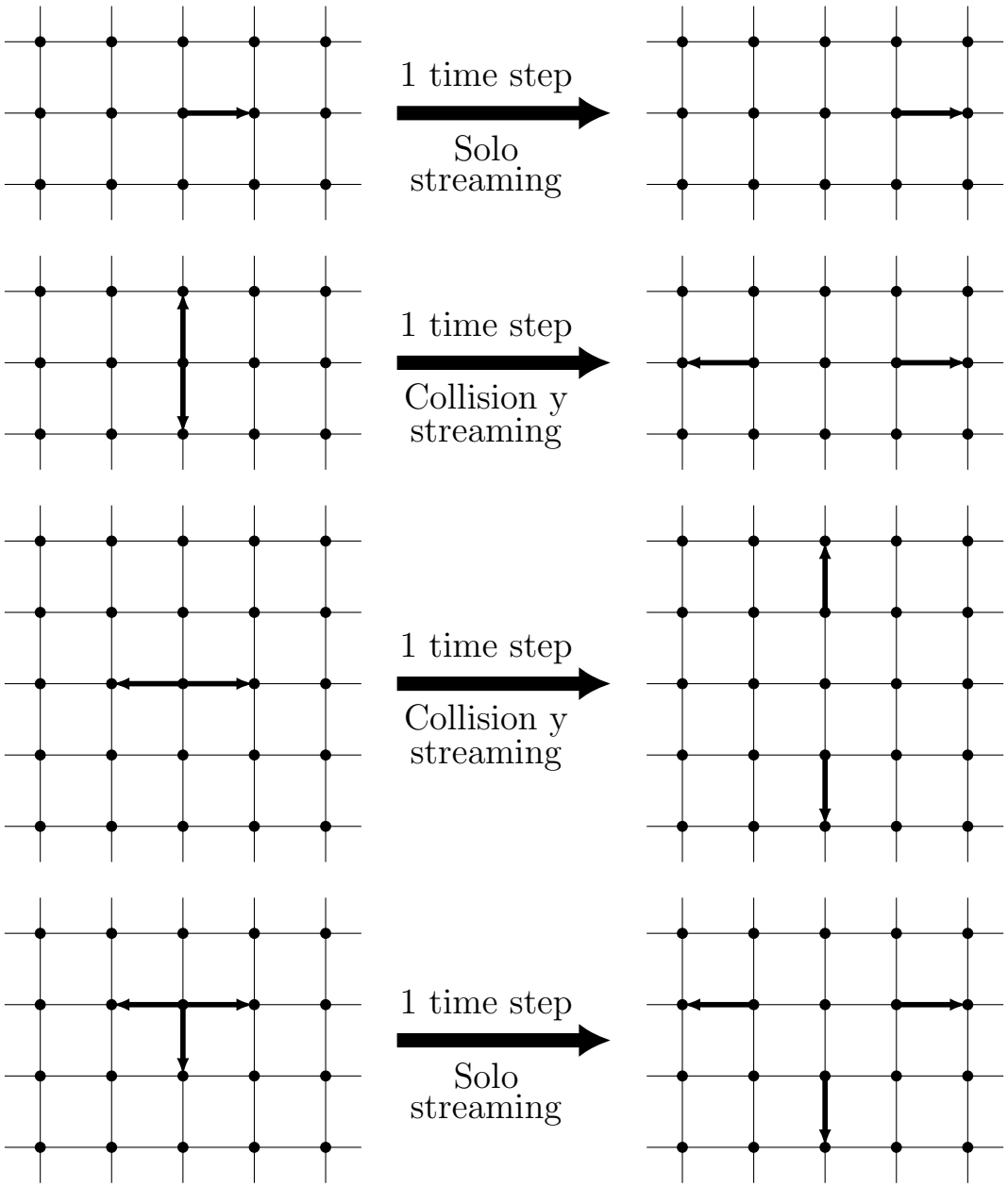

Figura 2: Evolución del método en un time step.
Solo son dibujadas las partículas existentes, i.e. $f_i = 1$.

2. Archivos de Entrada y Salida

Para el desarrollo de este laboratorio usted deberá descargar el script `generator.ipynb` publicado justo a este enunciado. Este código de python le ayudará a generar las mallas necesarias para este problema. Un archivo generado por este script contiene la distribución inicial de partículas con la cual debe comenzar su simulación. El formato de este archivo es:

- La primera línea contiene dos enteros N y M correspondientes a la dimensión (número de nodos) de la malla en la dirección y y x respectivamente.
- Las líneas 2, 3, 4 y 5 del archivo contienen cada una $N \times M$ valores binarios correspondientes a la presencia o ausencia de una partícula en una dirección de cada nodo. La i-ésima línea del archivo contiene los valores f_{i-2} (ver Figura 1), e.g. la línea 2 contiene los valores f_0 de cada nodo. Considere que la malla se representa mediante filas concatenadas, es decir, primero están los valores asociados a los M nodos con $y = 0$, después los M nodos con $y = 1$, y así sucesivamente hasta los M nodos con $y = N - 1$.

Como comprenderá al leer la sección Desarrollo, no es necesario generar archivos de salida para este laboratorio, pero sin duda alguna pueden ser una muy buena forma de ver que su código esté haciendo lo correcto. El script `generator.ipynb` no solo genera los archivos con la data inicial, sino también permite graficar las partículas en la malla, permitiendo un seguimiento visual del comportamiento esperado. Para lograr esto, los archivos de salida de su código deben presentar el mismo formato que los archivos de entrada. Es recomendable generar archivos pequeños (11×11 es un buen tamaño) para poder visualizar las partículas y detectar errores. También se recomienda modificar la aleatoriedad de las partículas generadas para provocar intencionalmente distintos casos de collision o de interacción con los bordes.

3. Desarrollo

Antes de comenzar, se le recomienda que para todas las implementaciones que realice en este laboratorio, compruebe que se conserva la masa del sistema:

$$\sum_{k=1}^{4}\sum_{j=1}^{N}\sum_{i=1}^{M} f_k((x_i, y_j), t_q) = \sum_{k=1}^{4}\sum_{j=1}^{N}\sum_{i=1}^{M} f_k((x_i, y_j), 0) = \text{constante} \qquad \forall q \in \{1, \ldots, 1000\}$$

Esto significa que para todos los tiempos simulados, el número de partículas en el dominio se mantiene constante, ya que no deberían desaparecer partículas ni aparecer nuevas. Basta con que compruebe para el tiempo final. Esto es únicamente para que usted pueda cerciorarse de que su código no presenta errores, pero no se exige en el informe.

Otra nota importante a considerar es que, si bien se le recomendó generar archivos de entrada pequeños para la visualización y prueba de su código, los tiempos requeridos en el informe deben ser medidos al trabajar con una malla de tamaño 2000×2000.

1. Cree dos funciones de CPU encargadas de leer el archivo `initial.txt`, una utilizando el enfoque Structure of Arrays y la otra usando Array of Structures, obteniendo en ambos casos un array de dimensión $N \times M \times 4$. Para cada una de ellas, implemente dos CUDA kernels (uno para el paso de collision y otro para el paso de streaming) para simular un time step del método, considerando en todos los casos que cada hebra se encarga de procesar un nodo de la malla. Simule 1000 time steps llamando a los kernels desarrollados en un ciclo for de 1000 iteraciones. Obtenga el tiempo que tardó la ejecución de este ciclo en cada caso.

   Para efectos de su implementación considere condiciones de borde periódicas, i.e. los nodos en el borde superior ($y = N - 1$) son vecinos de los nodos en el borde inferior ($y = 0$) y lo mismo ocurre para los nodos de las fronteras este ($x = M - 1$) y oeste ($x = 0$).

Hint: Recuerde que debe evitar las condiciones de carrera que pudiesen surgir en el paso de streaming, e.g. cuando una hebra intenta trasladar una de sus partículas a uno de sus vecinos sobreescribiendo su valor, pero la hebra encargada de dicho vecino aun no ha realizado este proceso.

[Informe] Presente una tabla o gráfico con los tiempos obtenidos para ambas implementaciones.

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

2. Elija la versión más eficiente de las dos implementadas y modifique el problema resuelto cambiando a condiciones de borde de muro. En este caso, las partículas que llegan a un nodo situado en el borde con una dirección ortogonal a dicho borde se reflejan o "rebotan" hacia el sentido contrario antes de realizar el streaming (ver Figura 3). Para esta implementación, omita el paso de collision en los nodos de borde, i.e. sí existen únicamente dos partículas moviéndose en dirección opuesta en el mismo nodo ubicado en la frontera, permita que se "atraviesen" y sigan su camino sin rotar en 90° como en el resto de los nodos. Compare la utilización de if-statements, operador ternario y multiplicación booleana en el manejo de los condicionales relacionados al paso de collision y las condiciones de borde (utilice el mismo método a la vez para verificar ambos tipos de condición). Reporte en una tabla los tiempos obtenidos para 1000 iteraciones con cada método.

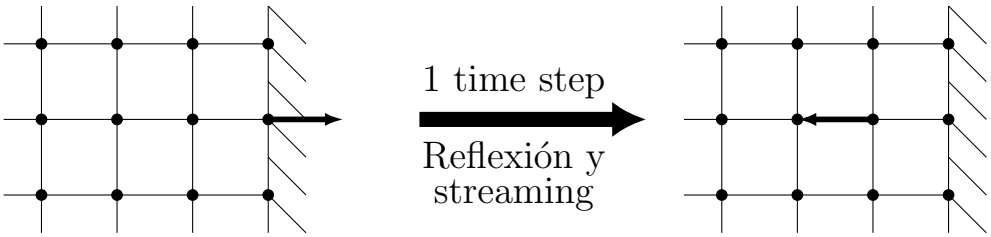

Figura 3: Condición de borde de muro.

[Pregunta] ¿Es alguno de los 3 métodos para manejar condiciones más eficiente que los demás? Comente el porqué de este resultado.

3. También considerando condiciones de borde de muro, en esta ocasión, desarrolle un único CUDA kernel que simule un time step del método, es decir, un único kernel que incluya los pasos de collision y streaming. Puede realizar el preprocesamiento que desee a los datos antes de comenzar el ciclo de 1000 iteraciones.

   Para la realización de este kernel utilice el método para manejar condiciones que obtuvo un mejor resultado en la pregunta anterior. Del mismo modo, siga utilizando la mejor organización de los datos encontrada en la pregunta 1. Mida e indique el tiempo que demoran 1000 iteraciones.

   [Pregunta] ¿Resultó su nueva implementación ser más eficiente? ¿Aumentó, redujo o mantuvo el número de accesos a memoria global respecto a sus implementaciones anteriores? Comente al respecto.

4. Reglas y Consideraciones

Entrega

- La entrega debe realizarse en un archivo de nombre Lab2-X.tar.gz (formatos rar y zip también son aceptados), donde X debe ser reemplazado por el número de su grupo. Diríjase a la inscripción de grupos para consultar su número.
- El archivo de entrega debe contener un informe en formato pdf junto con el código implementado para resolver el laboratorio. Se le ruega entregar un código ordenado.
- El informe debe contener:
  - Título y número del laboratorio.
  - 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, comentarios, observaciones o supuestos que surgieron durante el desarrollo del laboratorio.
- El descuento por día de retraso es de 30 puntos, con un máximo de 1 día de retraso. No se aceptarán entregas posteriores.
- 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 incumplimiento de estas reglas implica descuentos en su evaluación.
- La fecha de entrega es el día Martes 09 de Junio. Se habilitará la opción de entrega en Aula.

Consideraciones

- Dado que solo manejaremos ceros y unos, es libre de utilizar arreglos de tipo `unsigned char`.
- Para mediciones de tiempo, utilice `cudaEvent()` para códigos de GPU. Trabaje con milisegundos $[ms]$.
- En caso de optar por la realización de gráficos, puede optar por la herramienta que más le acomode. La librería matplotlib de Python siempre es una buena opción.
- Utilice 256 hebras por bloque en su implementacion.