Tarea 3 - Paralelización de Autómatas Celulares 3D

CC4102 - Diseño y Análisis de Algoritmos
Profesor: Pablo Barceló Auxiliar: Jorge Bahamonde

Fecha de Entrega: 14 de Agosto de 2015

1 Introducción

El objetivo de esta tarea es implementar un autómata celular 3D utilizando paralelismo y evaluar esta implementación.

Se espera que entregue el código desarrollado y un informe que indique claramente los siguientes puntos:

1. Las *hipótesis* escogidas antes de realizar los experimentos.
2. El *diseño experimental*, incluyendo los detalles de la implementación de los algoritmos, la generación de las instancias y las medidas de rendimiento utilizadas.
3. La *presentación de los resultados* en forma de una descripción textual, tablas y/o gráficos.
4. El *análisis e interpretación* de los resultados.

2 Autómatas 3D

Un autómata celular puede verse como una grilla regular de células, cada una en un estado particular (existiendo un número finito de éstos). Para cada célula puede definirse su *vecindad*, típicamente basándose en la geometría de la grilla. El estado global del autómata está dado por el conjunto de los estados de sus células. La evolución del autómata (es decir, obtener un nuevo estado global) ocurre de acuerdo a la aplicación de una *regla* preestablecida, que asigna a cada célula un nuevo estado dependiendo de los estados de ésta y de las células en su vecindad. La regla se aplica a todas las células de forma simultánea.

Geometría de la grilla

Un autómata celular 3D es un caso particular en el que la grilla es un arreglo tridimensional de células, por lo que cada célula posee 26 potenciales vecinos. En este caso, trabajaremos con una grilla periódica: el borde superior conecta con el inferior, el borde derecho conecta con el izquierdo, etc. Esto implica que todas las células tienen 26 vecinos.

Estados y Reglas

Por simplicidad, consideraremos células con sólo dos estados (prendida y apagada). Como regla, utilizaremos la siguiente:

- Una célula apagada se prende si su número n de vecinos prendidos cumple $n_1 \leq n < n_2$.
- Una célula prendida sólo se mantiene prendida si su número n de vecinos prendidos cumple $n_3 \leq n < n_4$.

De esta forma, los parámetros n_1, n_2, n_3, n_4 definen el funcionamiento del autómata.

Paralelización

Como la mayoría de los autómatas celulares, esta variante es fácilmente paralelizable. Un programa que calcula el próximo estado de parte del autómata puede hacerlo de forma *local*, examinando el estado anterior y sin preocuparse del siguiente estado de las demás células.

La idea básica de esta tarea es particionar el trabajo de calcular el siguiente estado del autómata en un número de piezas y luego asignar un *threads* a cada pieza. Cada *thread* recibe información de los vecinos de la pieza en que trabaja, calcula el nuevo estado de cada elemento en esta pieza y lo incorpora al nuevo estado global.

Utilizaremos dos formas de particionado. Para ambas consideraremos que las células están numeradas desde 0 hasta $C - 1$.

Cuociente

En este método, el j-ésimo thread (de un total de N threads) calculará el siguiente estado de cada célula cuyo índice i sea tal que

$$i \in \left[j \cdot \frac{C}{N}, (j+1) \cdot \frac{C}{N} \right) \text{(utilizando división entera)}.$$

Módulo

En este método, el j-ésimo thread calculará el siguiente estado de las células cuyo índice i es tal que $i \bmod N = j$.

Otros particionados

Si lo desea, puede plantear otro tipo de particionado y utilizarlo en vez de uno de los ya especificados, explicándolo de manera apropiada en su informe.

3 Hipótesis y Datos

Plantee una hipótesis que le resulte de interés, considerando las estrategias de particionado que utilice. Puede estudiar, además, reglas y/o condiciones iniciales que considere relevantes (condiciones iniciales aleatorias pueden ser un buen punto de partida), así como otras variaciones. Si tiene dudas sobre si las variaciones que desea introducir son válidas, no dude en consultar.

Puede estimar $T(n, 1)$ como el tiempo de ejecución promedio al utilizar sólo un thread o programar un algoritmo no paralelo que considere más cercano al óptimo. Considere las siguientes métricas al momento de generar su hipótesis:

- El tiempo de ejecución $T(n, p)$ (medido como tiempo real).
- El speedup $S(n, p)$.
- La eficiencia $E(n, p)$.

4 Experimentos e Instancias

Para poner a prueba su hipótesis, utilice grillas de $m \times m \times m$ células, con m en $\{5, ..., 24\}$ (al menos). En el caso de usar grillas no cúbicas (en particular, si su hipótesis lo requiriera), considere una cantidad de elementos equivalente. Utilice 2^k threads (particionando la grilla, por lo tanto, en 2^k piezas), con k en $\{1, ..., 4\}$ (al menos). Mida tiempo de ejecución, speedup y eficiencia para cada combinación de parámetros (independientemente de la hipótesis que haya escogido). Dado que obtendrá muchos datos, probablemente sea preferible mostrar gráficos u otras visualizaciones en su informe y dejar las tablas de datos en un apéndice de éste. Documente además las especificaciones de la máquina donde ejecutó sus experimentos (en particular, el número de procesadores).

5 Entrega de la Tarea

- La tarea puede realizarse en grupos de a lo más 2 personas.
- No se permiten atrasos.
- Para la implementación puede utilizar `C`, `C++`, `Java` o `Python`. **Importante:** si decidiera utilizar Python, **averigüe sobre el Global Interpreter Lock antes de comenzar** [1] y tome las medidas necesarias. Para el informe se recomienda utilizar LaTeX.
- Escriba un informe claro y conciso. Las ponderaciones del informe y la implementación en su nota final son las mismas.
- La entrega será a través de U-Cursos y deberá incluir el informe junto con el código fuente de la implementación (y todas las indicaciones necesarias para su ejecución).

[1] La siguiente página puede ser un buen punto de partida `https://wiki.python.org/moin/GlobalInterpreterLock`