Proyecto # 2: Modelo de Ising 2D

Paralelización de procesos con python-multiprocessing

Daniel Estrada Acevedo

Instituto de Física—Universidad de Antioquia

Resumen

En este trabajo se presenta una forma de paralelizar la simulación de Ising dos dimensional desarrollada en el primer proyecto del curso de Computación cientifica avanzada I [1], para ello se usa la clase `Pool` del módulo de Python `multiprocessing` para la ejecución en paralelo del algoritmo de *Metropolis-Hasting*. Acá se muestra una gráfica comparativa de los tiempos de ejecución del programa, en serial y en paralelo, en función del tamaño del sistema de partículas y se muestra que con esta simple optimización del código se logra acelerar el proceso en un factor de 3.5 usando 4 núcleos de procesamiento.

Palabras clave— Programación en Paralelo, Multiprocessing, Metropolis-Hasting, Modelo de Ising.

1. Introducción

La computación paralela es una técnica fundamental en el ámbito de investigación científica, especialmente en el campo de la simulación, donde se llevan a cabo cálculos y operaciones complejas que requieren de una gran capacidad de procesamiento. Aunque los conceptos de la computación en paralelo pueden ser bastante amplios y diversos, en el ámbito de la simulación científica es común enfocarse en la paralelización a nivel de tareas, en la cual la arquitectura de Von neumann se mantiene, pero se explota utilizando múltiples unidades de procesamiento que ejecuten las diferentes tareas que componen un proceso completo.[2]

De forma particular, en este trabajo se mostrará el poder de la computación en paralelo con una simulación de un modelo de Ising dos dimensional, que se caracteriza por requerir de tiempos de cómputo mayores cuanto más grande es el sistema. Para esto, se medirán los tiempos de ejecución para mayas cuadradas de diferentes tamaños y se analizará la mejora de rendimiento con respecto al desarrollo secuencial típico de la simulación. Se hará uso de la librería de Python `multiprocessing`, que representa una forma muy simple y poderosa de tener un primer acercamiento a la paralización de tareas por medio de la creación de procesos que pueden desarrollarse de forma simultánea.

En [1] se pudo presentar simulaciones para mayas de dimisiones 10×10 y 30×30, omitiendo sistemas más grandes por la gran cantidad de tiempo necesaria para correr cada una, sin embargo, acá se pretende usar la mejora de rendimiento alcanzada con la paralelización del programa para poder analizas mayas con tamaños más significativos.

2. Marco Teórico

Como ya se planteó en el primer proyecto de este curso [1], el modelo de Ising es propuesto para estudiar el comportamiento de materiales ferromagnéticos, y representa un problema paradigmático para la física estadística ya que que permite ilustrar los fenómenos de transición de fase de sistemas macroscópicos. [3]

En [1] además de plantear las bases conceptuales importantes sobre el modelo de Ising dos dimensional, se mostró, cómo a partir de la simulación de mayas cuadradas de espines evolucionadas bajo un algoritmo de *Metrópolis-Hastings*, se podía observar una transición de fase calculando el parámetro de orden M (magnetización del sistema) en función del parámetro $\beta = \frac{1}{k_B T}$ por medio de un estimado de Monte Carlo. Pero como se mencionó antes, los tiempos de computo deben mejorarse si se quieren analizar sistema con más partículas, por lo que, a continuación, se da una breve idea de en lo que consiste la paralización a nivel de tareas para aplicarlo a la simulación del modelo.

2.1. Paralelización de tareas

Para resolver un problema de forma computacional, típicamente, se plantean una serie de tareas o procesos que deben ser desarrollados. Cada una de estas tareas está compuesta de un conjunto de instrucciones que son entregadas a las unidades de procesamiento en el orden en el que el programador lo estableció, esta situación se ilustra en la parte derecha de la Figura 2. La solución al problema se obtiene al finalizar la ejecución de cada tarea y para esto se invierte cierta cantidad de tiempo de computo que se denotará como T_s. Ahora bien, los computadores modernos, por lo general y por fortuna, cuentan con más de una unidad de procesamiento, lo cual brinda al programador la posibilidad de repartir la carga

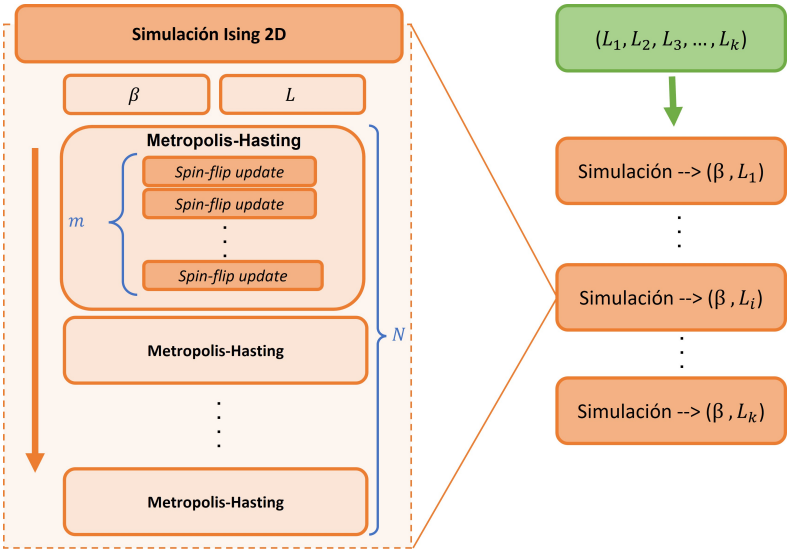

Figura 1: Representación gráfica del programa que ejecutas de forma secuencial las simulaciones del modelo de Ising para diferentes tamaños del sistema (`run.py`).

de trabajo del problema, y, por ende, estará de acuerdo con esta afirmación, la posibilidad de reducir el tiempo de ejecución ($T_p < T_s$). Cuando las tareas de un problema se ejecutan por unidades de procesamiento distintas, se puede decir entonces que se está desarrollando un proceso de computación en paralelo.

Idealmente, la ejecución sincrónica, como se ve a la izquierda Figura 2, es la que permite el tiempo más óptimo de procesamiento y es el objetivo general cuando se decide paralelizar un problema, sin embargo, no todas las tareas de un programa se pueden paralelizar, y esto dependerá de la correlación o dependencia entre los procesos, al final, solo cierto porcentaje del programa podrá ser paralelizado, pero será suficiente para mejorar en gran medida el rendimiento de un programa, simulación o análisis científico.

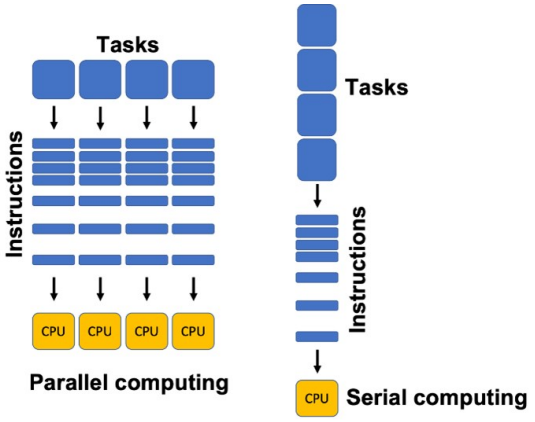

Figura 2: Representación gráfica de la paralelización de procesos a nivel de tareas.

2.2. Multiprocessing

La forma de implementar tareas en paralelo en un programa está lejos de ser única y depende en gran medida del lenguaje que se quiera usar. Una primera aproximación a este paradigma de la programación puede hacerse en Python a través del módulo `multiprocessing`, una librería bastante robusta que permite aprovechar al máximo los múltiples procesadores de una maquina usando para esto procesos. En particular, en este módulo se puede encontrar un objeto, conocido como `Pool`, que ofrece una forma de paralelizar la ejecución de funciones a través de la distribución simultanea de estas tareas con diferentes argumentos de entrada en los módulos de procesamiento disponibles. Para la simulación de este trabajo entonces, se hace uso de esta clase para optimizar el tiempo de ejecución del programa como se explicará en la metodología.

3. Metodología

La simulación del Modelo de Ising dos dimensional consiste en tomar una maya cuadrada de partículas y evolucionar la configuración de espines de acuerdo con la teoría de colectivos canónicos de física estadística siguiendo para esto un algoritmo de MCMC (*Markov chain Monte Carlo*), en específico el algoritmo de *Metropolis-Hasting*. Los detalles de esto ya fueron detallados en [1], pero se puede resumir las tareas del programa como se muestra en el recuadro titulado "Simulación Ising 2D" de la Figura 1.

Como se puede ver, un proceso de simulación consiste básicamente en: Dado un tamaño de maya (L) y la temperatura del sistema (β), se corre N veces la tarea etiquetada como "Metropolis-Hasting", cuya ejecución lleva una configuración inicial de espines (que puede ser ordenada o desordenada), en m pasos del algoritmo, a una final de las 2^{L^2} posibles configuraciones del sistema de acuerdo a la distribución de probabilidad del modelo. [1]

Como se mencionó al comienzo, uno de los objetivos es analizar el tiempo gastado en la ejecución de las simula-

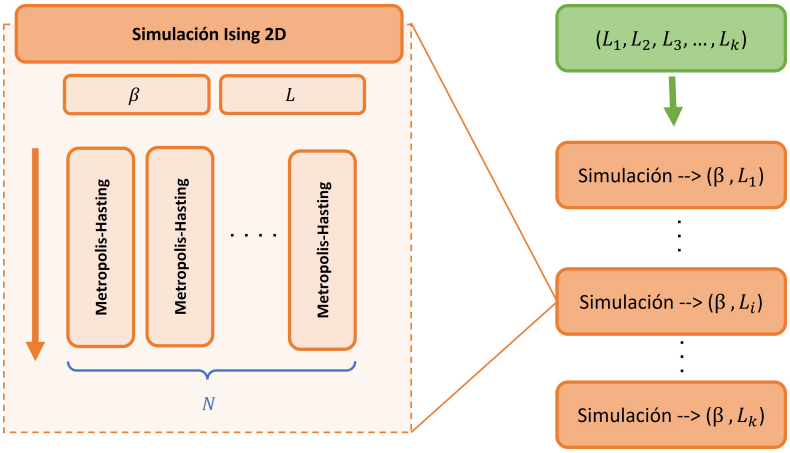

Figura 3: Representación gráfica del programa que ejecutas las simulaciones paralelizadas del modelo de Ising para diferentes tamaños del sistema (`run_p.py`).

ciones, entonces, la idea es correr para diferentes tamaños y ver el comportamiento de T_s vs L, donde T_s es el tiempo que toma la ejecución de "Simulación Ising 2d" de forma serial, ósea sin ser paralelizada. Por otro lado, también se analizará T_p vs L, donde T_p es el tiempo de ejecución pera de la versión paralelizada del problema, que consiste en las tareas que se representan gráficamente en la Figura 3.

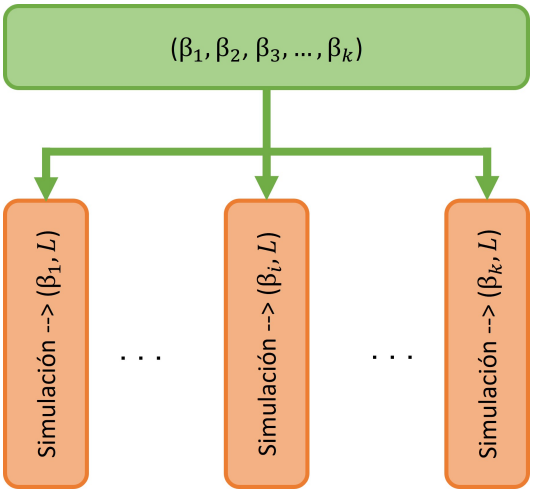

Figura 4: Representación gráfica del programa que ejecuta en paralelo las simulaciones del modelo de Ising para diferentes valores de β (`run_transition_p.py`).

Se resalta el concepto de paralelización de tareas del que se hablaba antes, note de la Figura 3 que el problema se optimizó ejecutando simultáneamente los N procesos de "Metropolis-Hasting" que no tienen correlación entre ellos.

Para el otro propósito de este trabajo, correr una simulación de mayor tamaño, se plantea entonces la ejecución de un programa paralelizado con la estructura que se plasma en el diagrama de la Figura 4, allí se ilustra la ejecución en paralelo de diferentes simulaciones para varios valores de temperatura, con lo cual se espera obtener un diagrama que muestre la transición de fase a través de la magnetización del sistema. Cabe resaltar que en este último esquema de paralelización, el proceso de simulación no corresponde con el proceso paralelizado que se planteo en la Figura 3, sino con el proceso secuencial de ejecutar N veces el algoritmo de Metropolis de la Figura 1.

4. Resultados

La paralelización del proceso "Simulación Ising 2d" que se planteo en la Figura 3, representó una mejora bastante buena en el tiempo de ejecución, pudiendo llevar una tarea que tardaba alrededor de 5 horas para una maya grande de 480×480, a un poco menos de un par de horas. En la Figura 5 se presenta gráficamente este resultado, allí, la curva roja representa el incremento en el tiempo de ejecución para el programa sin paralelizar y en azul se presentan los tiempos del programa ya optimizado.

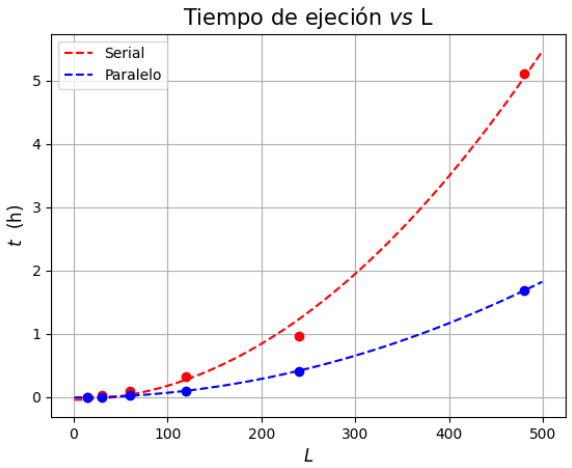

Figura 5: Gráfica de los tiempos de ejecución de los programas para diferentes tamaños del sistema. En rojo, los tiempos de la ejecución en serial, en azul, los tiempos del programa paralelizado.

De los datos de tiempo se pudo promediar las fracciones $\frac{Ts(L)}{Tp(L)}$, obteniendo un valor de 3.5 ± 0.7, que representa

una cantidad conocida como aceleración observada (*Observed Speedup*). [2]

Ahora, el propósito de correr mayas grandes usando la paralelización planteada en la Figura 4 no presentó el rendimiento esperado, aunque si se mejoró con respecto a tratar de correrlo en serial. Si se analiza con detalle el proceso que ese programa conlleva, podrá estar de acuerdo con que el tiempo de cómputo sigue siento alto, esto, porque con base en la Figura 5, una simulación de $250x250$ partículas tardaría alrededor de 1.5h, y con un esquema ideal como el de la Figura 4, se podría correr todos los β en simultaneo en menos de 2 horas, sin embargo, la situación está lejos de ser la idea, no es posible que todos los procesos para cada β diferente se ejecuten al tiempo, pues el número de procesadores es finito y no lo suficientemente grande para abarcar los casi 100 procesos deseados ($\beta = \{0.01, 0.02, 0.03, \ldots, 1\}$), en su lugar, con una maquina que tiene a su disposición 4 núcleos de procesamiento, tomaría $(\frac{100}{4} \times 1.5)$h terminar el proceso de simulación deseado. Por tanto, sin más tiempo para correr esta simulación, se decide presentar en este trabajo el análisis respecto a los procesos de paralelización solamente y se omite el análisis de transición de fase del sistema.

A pesar de no haber podido generar una simulación de una maya grande, se opto por correr los mismo tamaños de [1], con lo que se obtuvo una mejora en el valor del parámetro crítico, recuperando los valores $\beta_c = 0.56$ y $\beta_c = 0.66$ para las mayas de $L = 10$ y $L = 30$ respectivamente, lo cual, representa un reducción del error con respecto a lo presentado antes de casi 17 % para la medida de 0.56 por ejemplo.

Los códigos desarrollados para este trabajo están publicados en Github.

5. Conclusiones

- De forma satisfactoria se pudo lograr una optimización del programa que desarrolla la simulación del modelo de Ising 2D, se logró correr el programa en localmente con 4 núcleos, y se pudo alcanzar un rendimiento más de 3 veces superior.
- La paralelización con `multiprocessing` es una excelente alternativa para optimización a nivel de tareas, sin embargo, presenta algunas dificultades a la hora de hacer procesos en paralelo de forma anidada, para entender este problema, note que la paralelización planteada en la Figura 4 utiliza los procesos de simulación seriales y no los paralelizados de la Figura 3, ya qué no hubo forma de encapsular en una función la ejecución de los algoritmos de *Metropolis-Hasting* de forma paralela, esto, porque la librería tiene algunas restricciones en cuanto al uso de sus objetos por fuera de la función `main()` del programa. Este es un asunto a estudiar más a fondo para poder lograr una paralelización total del programa y llegar a reducir al máximo los tiempos de computo.
- EL uso de un cluster podría se una solución al problema que se planteo en el item anterior, La idea sería enviar como tareas individuales a los nodos del cluster cada simulación con un valor de β diferente, en donde ahora cada programa si correría la evolución del sistema de forma paralela como en la Figura 3. Sin embargo, esta opción no se implementó en este trabajo porque no se pudo resolver la forma de crear un programa que enviara los diferentes trabajos mencionados variando el parámetro de temperara, y hacer esta labor a mano no era una alternativa. Queda pendiente entonces encontrar la forma de implementar lo mencionado. Note que esta forma de optimización es un poco burda, porque no se tiene certeza de la simultaneidad de la ejecución de los programas en los nodos del cluster, lo que al menos se garantiza es que los tiempos de simulación de cada programa será menor por ejecutar los algoritmos de Metrópolis en paralelo.

Referencias

[1] Daniel Estrada. *Proyecto # 1: Modelo de Ising 2D : Aplicación de los Métodos de Monte Carlo en física*. Oct. de 2021. URL: https://github.com/DanielEstrada971102/2D_IsingSImulation/blob/main/Reporte_escrito.pdf.

[2] Manuel Sanchez, Cristian Garcia y Francisco Navarro. *La computación paralela: alta capacidad de procesamiento - Teldat Blog - Connectando el mundo*. Jul. de 2020. URL: https://www.teldat.com/blog/es/computacion-paralela-capacidad-procesamiento/.

[3] Wikipedia contributors. *Ising model — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-October-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Ising_model&oldid=1048384303.