Tarea 3 IMT2112

Pablo Rademacher

7 de noviembre de 2020

¿Se puede usar CG para resolver el problema?

La respuesta a esta pregunta es SI, y podemos ver esto gracias al teorema de Gershgorin: La matriz resultante va a tener, en cada fila, a lo más 5 elementos, uno de los cuales estará en la diagonal si o si. Este elemento (el centro de cada círculo) será:

$$C_{i,j} = \frac{\alpha_{i-\frac{1}{2},j} + \alpha_{i+\frac{1}{2},j}}{h_x^2} + \frac{\alpha_{i,j-\frac{1}{2}} + \alpha_{i,j+\frac{1}{2}}}{h_y^2} > 0$$

donde i y j son el índice del nodo representado por esta columna.
Además observemos que el radio $R_{i,j}$ (la suma del resto de la fila) cumple:

$$R_{i,j} \leq C_{i,j} - \left(\frac{\alpha_{i-\frac{1}{2},j} + \alpha_{i+\frac{1}{2},j}}{h_x^2} + \frac{\alpha_{i,j-\frac{1}{2}} + \alpha_{i,j+\frac{1}{2}}}{h_y^2} \right)$$

Observemos que además, como la matriz resultante será simetrica, todos sus valores propios serán reales, y el menor valor que podrán obtener es 1, ya que ningún círculo cubrirá algun número menor a este (este 1 surge de restarle el radio al centro de cada círculo). Como todos sus valores son mayores a 1, entonces la matriz es difinida positiva, asi que se le puede aplicar Gradientes Conjugados.

Implementación paralela:

En esta tarea usé el formato "stencil"para guardar los elementos de A. Para ello, cree 5 "matrices"(N, W, C, E, S) en las cuales, en su componente i, j, está guardado el elemento que, al realizar $Ax = b$, estaría multiplicando el nodo superior, a la izquierda, el mismo, a la derecha o inferior, respectivamente, del dominio x. De esta manera, para obtener $b_{i,j}$, hago:

$$b_{i,j} = N_{i,j}x_{i+1,j} + W_{i,j}x_{i,j-1} + C_{i,j}x_{i,j} + E_{i,j}x_{i,j+1} + Si, jx_{i-1,j}$$

Todas estas representaciones de matrices y vectores como dominio son del mismo tamaño $(N_x \times N_y)$, y estan repartidas por partes ïguales.$^{\text{en}}$tre todos los procesadores. Use una partición por filas ya que esto me ayuda a realizar buena parte de los productos punto de manera local (despues es un simple allreduce) y en los productos matvec lo unico que hay que comunicar es una fila a cada procesador çontiguo", la cual esta toda junta en memoria.

La manera en que hago estos envios esta explicada en el codigo, pero resumiendo, primero cada procesador le entrega su ultima fila al siguiente. El ultimo pasa directamente a recibir, y esto genera una çadena de recibimientos". Ahora repito esto mismo pero con las primeras filas, que son enviadas al procesador anterior. Si bien no es lo más eficiente, sí se logra el cometido y con un codigo relativamente sencillo, además de ser versatil ante la paridad de la cantidad de procesadores totales.

readme

El funcionamiento del codigo esta bastante explicado como comentarios, asi que no voy a ahondar en eso. Lo que sí, tengo un par de disclaimers:

1. El codigo corre para cualquier cantidad de nodos y/o procesadores.

2. Por alguna razon si trato de correr el archivo con el job.sh, me manda error. Sin embargo, se puede correr compilando con ‘mpic++ main.cpp -std=c++11‘ directamente en el cluster, y despues ejecutando con ‘mpirun -np p a.out‘, donde p es la cantidad de procesadores.

3. Al aumentar el numero de iteraciones todas las normas del residual empiezan a ser nan. Esto se evita si se tiene un numero razonable (< 1000) de iteraciones.

4. Puedes modificar la función f y va a seguir convergiendo.

5. Creo que perdi como 6 horas buscando errores que resultaron ser estupidos o que se arreglaron solos.