UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

Taller - Sistemas Distribuidos

Docente: Carlos Andrés Gómez Vasco

IMPORTANTE: Resuelva, dejando claro sus cálculos paso a paso y sin omitir detalle alguno. Realice los diagramas utilizando las herramientas necesarias para su construcción. Incluso, de ser necesario, use colores para denotar diferencias en las situaciones a resolver.

Nombre: _____ Grupo: _____

EJERCICIOS

(5.0 Puntos)

1. En este taller se propone el estudio numérico y la optimización computacional de un problema clásico de la física matemática: la resolución de la ecuación de Poisson bidimensional asociada al potencial electrostático en una placa conductora cuadrada. El problema se formula en el dominio $(x, y) \in [0, 1] \times [0, 1]$, con condiciones de frontera de Dirichlet homogéneas, es decir, $\phi = 0$ en todo el borde del dominio. La ecuación diferencial a resolver es $\nabla^2\phi(x, y) = f(x, y)$, donde la fuente está dada por $f(x, y) = \sin(\pi x) \sin(\pi y)$.

   El dominio deberá discretizarse mediante una malla uniforme de tamaño $N \times N$, empleando aproximaciones en diferencias finitas centradas para el operador Laplaciano. A partir de esta discretización se obtendrá un esquema iterativo tipo Jacobi, cuya expresión discreta deberá implementarse explícitamente en el código. El método deberá ejecutarse hasta que la norma infinito de la diferencia entre iteraciones consecutivas sea menor que 10^{-6}. Se deberá escribir claramente la ecuación discreta implementada y justificar el criterio de convergencia utilizado.

   La implementación deberá realizarse tanto en C++ como en Fortran, sin utilizar librerías externas, BLAS ni paralelización en esta etapa inicial.

   En C++ deberán implementarse tres variantes distintas de almacenamiento de la matriz bidimensional:

   a) Usando punteros a punteros (double**).

   b) Usando un bloque contiguo de memoria asignado dinámicamente (new double[N x N).

   c) Utilizando std::vector¡double¿como estructura contigua.

   En Fortran se empleará un arreglo bidimensional estándar real(8), dimension(N,N). En todos los casos deberá medirse el tiempo total de ejecución y el número de iteraciones requeridas para alcanzar la convergencia.

   Una vez obtenida la versión funcional básica, se procederá al análisis de localidad espacial y comportamiento de caché. Para ello, se deberán experimentar con el orden de los bucles anidados que recorren la malla. En C++ se comparará el desempeño de recorrer primero la dimensión asociada a filas y luego columnas, frente al orden inverso. En Fortran deberá realizarse el mismo análisis, teniendo en cuenta que el almacenamiento es column-major. Se deberán ejecutar experimentos para distintos tamaños de malla, al menos $N = 512, 1024, 2048, 4096$, y analizar cómo cambia el tiempo de ejecución al modificar el orden de acceso en memoria. El objetivo es identificar empíricamente el impacto de la localidad espacial y la organización interna del almacenamiento.

Posteriormente, se deberá implementar una versión optimizada mediante técnica de blocking o tiling, dividiendo la malla en subbloques de tamaño $B \times B$. Se evaluarán al menos cuatro tamaños de bloque distintos (por ejemplo, B=8,16,32,64), comparando el tiempo total de ejecución en cada caso. Se deberá relacionar el tamaño de bloque óptimo con las características de la jerarquía de memoria de su procesador (tamaño de caché L1 y L2), explicando por qué ciertos tamaños producen mejoras significativas y otros no.

Para cada implementación (naive y bloqueada, en ambos lenguajes), deberá reportarse el tiempo total, el tiempo promedio por iteración y el número total de iteraciones. Además, se deberá estimar el número de operaciones aritméticas por punto de la malla, el volumen aproximado de datos transferidos en memoria y la intensidad aritmética del algoritmo. Con base en este análisis, el estudiante deberá argumentar si el problema es predominantemente memory-bound o compute-bound, justificando su respuesta a la luz del modelo Roofline.

Finalmente, se deberá realizar una comparación crítica entre C++ y Fortran en términos de desempeño, facilidad de optimización y comportamiento de memoria. El informe deberá discutir explícitamente cómo influye el layout de memoria (row-major vs column-major), si el compilador logró vectorización automática y qué estructura de almacenamiento resultó más eficiente. No se aceptarán conclusiones descriptivas; todas las afirmaciones deberán estar sustentadas en datos experimentales obtenidos durante el taller.

Como entregable, se deberá presentar el código fuente completo en ambos lenguajes y un informe técnico estructurado que incluya la derivación matemática del esquema numérico, el análisis detallado del comportamiento de caché, tablas comparativas de rendimiento y conclusiones fundamentadas desde la perspectiva de arquitectura de computadores y cómputo científico de alto desempeño específicamente las técnicas de optimización .

El propósito de este taller no es únicamente resolver un problema numérico de física, sino demostrar comprensión profunda de la jerarquía de memoria, del diseño cache-aware de estructuras de datos y de la relación entre formulación matemática y eficiencia computacional en arquitecturas modernas.

(5.0 puntos)