Plan de Proyecto: OpenMP-MPI

Técnicas de Computación Científica

2024/25

Autores: Luna Asegurado Sastre y David Morilla Sorlí

POLITÉCNICA

ETS INGENIEROS INFORMÁTICOS

Índice

1 Definición y Esquema de Trabajo

1.1 Tema elegido

"*Tratamiento paralelo de imágenes usando OpenMP y MPI para la aplicación de filtros.*"

El problema a resolver es el tratamiento de imágenes mediante la aplicación de filtros de manera paralela utilizando OpenMP y MPI. En particular, buscamos procesar imágenes digitales a color, las cuales se pueden representar como matrices tridimensionales, de alto por largo por cada uno de los canales de colores, 3 en el caso de las imágenes RGB. El objetivo es transformar o mejorar las imágenes aplicando filtros como B&W, desenfoque, filtro de color, sepia, o mayor saturación. Se optimizará el rendimiento mediante el uso de programación paralela en dos enfoques: memoria compartida (OpenMP) y memoria distribuida (MPI). Finalmente se hará una versión que haga uso de ambos enfoques a la vez para intentar sacar el máximo rendimiento posible.

Este problema es adecuado para el trabajo porque conecta directamente con los conceptos de programación paralela estudiados en la asignatura. Por un lado, OpenMP permite abordar el paralelismo dentro de un único equipo con múltiples núcleos, aprovechando la memoria compartida. Por otro lado, MPI habilita el procesamiento en sistemas distribuidos, lo que es especialmente relevante para tareas que requieren un gran volumen de cómputo o datos. Al tratar imágenes como matrices tridimensionales, es posible dividir el procesamiento en tareas independientes, logrando una mejora significativa en el tiempo de ejecución respecto a enfoques secuenciales.

Como objetivos del trabajo esperamos implementar filtros básicos de procesamiento de imágenes usando OpenMP y MPI, poder comparar el procesamiento paralelo entre OpenMP y MPI, así como poder analizar cómo factores como el número de hilos, nodos y el tamaño de las imágenes impactan en la eficiencia y la escalabilidad.

Esperamos demostrar que tanto OpenMP como MPI mejoran significativamente el rendimiento respecto al procesamiento secuencial. Asimismo, se buscará identificar las ventajas y desventajas de cada enfoque según el contexto de uso, ya sea en sistemas con múltiples núcleos o entornos distribuidos.

1.2 Índice preliminar del proyecto

1. Introducción
   - 1.1. Contexto y motivación
   - 1.2. Objetivo del trabajo
2. Marco teórico
   - 2.1. Representación de imágenes como matrices
   - 2.2. Fundamentos de OpenMP y MPI
3. Punto de partida y profiling
   - 3.1. Descripción de los filtros a aplicar
   - 3.2. Diseño e implementación del código secuencial
   - 3.3. Búsqueda de los hotspots del programa
4. Implementación
   - 4.1. Diseño e implementación del código OpenMP y MPI y ambos.
5. Resultados y análisis
   - 5.1. Comparativa entre procesamiento secuencial, OpenMP y MPI y ambos.

5.2. Escalabilidad y eficiencia de cada enfoque
6. Conclusiones
   6.1. Resumen de hallazgos
   6.2. Limitaciones y propuestas de mejora

1.3 Pruebas a realizar

- Procesar imágenes de diferentes resoluciones (por ejemplo, 640×480, 1920×1080 y 3840×2160) con varios filtros.
- Evaluar el tiempo de ejecución en secuencial, OpenMP y MPI, y ambos variando:
  - Número de hilos en OpenMP.
  - Número de nodos en MPI.
- Analizar la escalabilidad y eficiencia comparando resultados entre OpenMP (en un único equipo con múltiples núcleos) y MPI (en entornos distribuidos o simulados) y mostrar la máxima capacidad de optimización tras juntar ambas tecnologías.

1.4 Herramientas a utilizar

En el desarrollo del proyecto, utilizaremos una combinación de hardware, software y librerías específicas que nos permitirán implementar, ejecutar y analizar el rendimiento de las soluciones desarrolladas en paralelo. Estas herramientas han sido seleccionadas cuidadosamente para garantizar la compatibilidad con las tecnologías OpenMP y MPI, así como para facilitar el procesamiento de imágenes y la evaluación del rendimiento.

1.4.1 Hardware y entorno de ejecución

- **Ordenadores portátiles personales**: Serán utilizados para el desarrollo y pruebas iniciales, configurados con compiladores y entornos compatibles con OpenMP y MPI.
- **Cluster Triqui**: Entorno distribuido que permitirá probar el rendimiento de las soluciones basadas en MPI en sistemas de memoria distribuida, simulando un entorno real de computación distribuida.

1.4.2 Lenguaje y tecnologías principales

- **Lenguaje C**: Utilizado para el desarrollo del código debido a su compatibilidad directa con OpenMP y MPI y su eficiencia en operaciones de bajo nivel.
- **OpenMP**: Tecnología para el paralelismo en memoria compartida, que permite dividir el procesamiento de las imágenes en múltiples hilos dentro de un mismo equipo.
- **MPI (Message Passing Interface)**: Framework para la programación en memoria distribuida, que facilitará la comunicación y coordinación entre nodos del cluster.
- **Uso combinado de OpenMP y MPI**: Exploraremos una solución híbrida que aproveche las fortalezas de ambos enfoques.

1.4.3 Compiladores y herramientas de desarrollo

- **GCC (GNU Compiler Collection)**: Compilador ampliamente utilizado que soporta tanto OpenMP como MPI.
- **MPICH/OpenMPI**: Implementaciones de MPI que se instalarán para la ejecución de código en entornos distribuidos.

1.4.4 Librerías específicas para procesamiento de imágenes

- **OpenCV**: Librería de código abierto ampliamente utilizada para el procesamiento de imágenes. Proporciona herramientas para cargar, manipular y guardar imágenes, facilitando la implementación de los filtros seleccionados.
- **libjpeg y libpng**: Librerías especializadas para el manejo de formatos de imagen como JPEG y PNG, optimizando la lectura y escritura de archivos.

1.4.5 Herramientas de análisis de rendimiento

- **Intel VTune Profiler:** Herramienta avanzada para analizar el rendimiento de aplicaciones paralelas, incluyendo métricas detalladas sobre el uso de memoria y CPU.
- **perf:** Utilidad ligera para analizar eventos de hardware y software, útil para identificar problemas en implementaciones paralelas.
- **gprof**: Herramienta de análisis de rendimiento para identificar "hotspots" en el código secuencial y paralelizado.
- **valgrind (callgrind)**: Para realizar un profiling detallado del consumo de recursos y detectar posibles ineficiencias.
- **time**: Comando básico para medir tiempos de ejecución de forma sencilla.
- **MPI profiling tools**: Herramientas específicas como **mpitop** o **MPI Performance Checker** para analizar la eficiencia de la comunicación en MPI.

1.4.6 Otros entornos y utilidades

- **Makefile**: Para la automatización del proceso de compilación y ejecución de pruebas con distintas configuraciones de hilos y nodos.
- **Git/GitHub**: Para la gestión del control de versiones y la colaboración entre los autores del proyecto.

Bibliografía:

Intel. *VTune Profiler Tutorials*. https://www.intel.com/content/www/us/en/developer/articles/training/vtune-profiler-tutorials.html?wapkw=vtune

Perf Wiki. *Perf: Main Documentation*. https://perfwiki.github.io/main/

Valgrind. *Callgrind User Manual*. https://valgrind.org/docs/manual/cl-manual.html

OpenCV. *Image Processing Module Documentation*. https://docs.opencv.org/3.4/df/d4e/group__imgproc__c.html

Peinado, J. *Tratamiento de imágenes en paralelo*. https://personales.upv.es/jpeinado/data/cpa/openmp/p1/p1_sesion2%20(imagenes).pdf

Aguirre, L. (2018). *OpenMP y la programación paralela*. https://medium.com/@leonardoaguirre_97179/openmp-y-la-programación-paralela-8990f14b95f3