Programación de Arquitecturas Emergentes [G4012452] [2024/2025]

Lab 3 - Programación básica y Programación de algoritmos paralelos con OpenMP

Resumen

El objetivo es adquirir los conocimientos básicos para la realización de programas paralelos en sistemas de memoria compartida y aplicar las metodologías de programación paralela vistas en clase utilizando OpenMP.

1. Lab 3.1 - Bloque 1: Programación básica con OpenMP

1. Averigua como se reparten las iteraciones (*scheduling* y *chunk size*) de un bucle for por defecto en un nodo del FTIII con el compilador GNU (*module load gcc*) y con el compilador de Intel (*module load icc*).
2. Estudia los siguientes ejemplos del documento *OpenMP Application Program Interface Examples (version 4.5.0)* disponible en el campus virtual:
   - The `omp_set_dynamic` and `omp_set_num_threads` Routines
   - The `omp_get_num_threads` Routine
   - The `reduction` Clause
   - The `collapse` Clause and The `linear` Clause
   - The `task` and `taskloop` Constructs
3. Inicializa una matriz con números secuenciales de la forma `fila*N+columna` de tamaño $M \times N =$ `1GiB`:
   - Reparte las iteraciones entre los hilos usando una construcción `parallel omp for`. Prueba a repartir las iteraciones en bloques de distinto tamaño y asignando distintas políticas de reparto de iteraciones.
   - Resuelve el mismo problema usando una construcción `parallel omp task` y `parallel omp taskloop` y compara el rendimiento de ambas con el mejor reparto del apartado anterior.
4. Repasa la organización del procesador con el comando `lstopo` visto en el Lab 1.

2. Lab 3.2 - Bloque 2: Programación de algoritmos paralelos con OpenMP

1. Implementa el **algoritmo de distancia euclídea** propuesto en el Lab 1.
   - Usando la cláusula `reduction` en OpenMP.

- Implementa tu propia versión de reducción y compara los resultados.
- Comprueba el *tipo de escalabilidad* (¿fuerte o débil?).

2. Implementa el **algoritmo de convolución** propuesto en el Lab 1 y determina la aceleración (*speedup*) usando diferente número de hilos.

3. Especificaciones

1. Usa un nodo interactivo durante el desarrollo y calcula el tiempo de ejecución (`wall time`) como una media de 10 ejecuciones usando un **nodo con 64 cores** cuando el programa esté preparado y libre de errores. Usa la cola para enviar procesos.
2. Calcula el **speedup** usando 2, 4, 8, 16, 32, 48 y 64 hilos.
3. Compila los programas con optimizaciones `-O2` y calcula el speedup respecto a la mejor versión secuencial. Ver *Hall of Fame* disponible en el campus virtual.
4. Separa en el tiempo de ejecución todo el overhead relacionado con gestión de memoria en CPU (malloc, free, inicialización, ...) que necesites para resolver el problema de forma paralela.
5. Explora diferentes reparto de iteraciones (*scheduling*) y tamaños de bloque (*chunksize*).
6. Comprueba que las implementaciones paralelas son correctas comparando los resultados con la versión secuencial.

4. Formato y fecha de entrega

1. Sube al campus virtual en un archivo comprimido `lab3.2.zip` los ejercicios del Lab 3.2 junto con un informe breve y detallado en formato pdf. Antes de subirlo, confirma con el profesor que cumples los requisitos.

   a) El código no puede tener comentarios.
   b) El informe debe presentar los resultados de forma clara y resumida mediante gráficas que muestren la aceleración y la escalabilidadg global y sólo de los kernels, entendiendo por 'kernel' la función que ejecuta el algoritmo de la distancia euclídea o la convolución.
   c) El informe no tiene que explicar el código implementado pero si cuestiones relacionadas con la arquitectura, patrones de concurrencia, discusión de los resultados, observaciones, conclusiones propias y toma de decisiones.
   d) Los datos numéricos deben guardarse por si fuera necesario una consulta por parte del profesor.

> ☞ **Fecha límite de entrega**: 14 de abril de 2025, a las 9:00 horas.

5. Evaluación

1. La evaluación se hará en base a las implementaciones paralelas, uso correcto de las directivas `OMP`, reparto de trabajo entre hilos, y a la calidad y defensa del informe.
2. Esta práctica tiene un **peso de 1.5 puntos en la nota final** y se evalúa sobre 10 puntos (5 puntos código, 5 puntos informe).