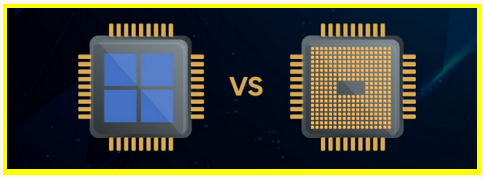
2024, INFO188 Tarea 2: Batalla de sorting paralelo

Profesor: Cristóbal A. Navarro**, Ayudante:** Alejandro Villagrán



En esta tarea, usted investigara sobre algoritmos de ordenamiento, comparará uno en CPU paralelo contra otro en GPU paralelo. Las actividades son:

1. Investigue el estado del arte de **algoritmos paralelos de ordenamiento de enteros en CPU y GPU** y en base a sus hallazgos elija el más rápido para cada chip. Debe fundamentar su elección en base a literatura de libros/artículos científicos y citarlos.
2. En un mismo programa de tipo ***.cu**, implemente:
    a. El algoritmo de CPU usando OpenMP.
    b. El algoritmo de GPU usando CUDA.
3. Su programa debe usar `Makefile` y debe ejecutarse como: `./prog <n> <modo> <nt>`
    a. **<n>** : tamaño del array de enteros generados con distribución aleatoria uniforme.
    b. **<modo>** : CPU -> 0, GPU -> 1
    c. **<nt>** : Numero de CPU threads (aplica para el modo CPU)
    d. Ejemplo CPU: `./prog $((2**26)) 0 8`
4. El programa debe retornar el tiempo (segundos con decimales) que se tardó en ordenar. Para medir tiempo puede usar los timers de OpenMP que sirven para ambos modos.
5. Debe realizar un benchmark que compare ambos algoritmos, debe obtener los siguientes gráficos (OJO: Cada gráfico muestra las dos curvas (CPU, GPU), no separar)
    a. **Tiempo (y) vs n (x):** El rango de **n** debe ir hasta lo más alto posible antes de que usted note que va a demorar mucho tiempo el benchmark. Como referencia, su rango de **n** debe comenzar bajo, como 1000 números, y llegar a los cientos de millones de números aleatorios. Cada punto debe ser un tiempo promedio de varias ejecuciones con los mismos argumentos. Escoja una cantidad de puntos suficiente para que la curva pueda verse suave.
    b. **[Solo CPU] Speedup y vs num-threads (x):** Debe fijar el problema en un **n** suficientemente alto, y calcular el speedup del algoritmo CPU usando x threads con respecto a su misma versión de 1 thread. El eje x debe cubrir el rango de la cantidad de núcleos de la CPU, el cual suele ser bajo.
    c. **[Solo CPU] Eficiencia paralela y vs num-threads (x):** análogo al anterior pero eficiencia paralela en vez de speedup.
    d. **[Solo GPU] Speedup y vs num-bloques (x):** Debe fijar el problema en un **n** suficientemente alto, y calcular el speedup del algoritmo GPU usando x bloques CUDA con respecto a su misma versión de 1 bloque CUDA. El eje x debe cubrir el rango de la cantidad de núcleos de la GPU, el cual es mucho más alto que el de la CPU.
    e. **[Solo GPU] Eficiencia paralela y vs num-bloques (x):** análogo al anterior pero eficiencia paralela en vez de speedup.
    f. **Speedup vs n**: Análogo al punto a), pero calculando la aceleración (speedup) con respecto al algoritmo `std::sort` de la librería estándar STL de C/C++.

IMPORTANTE:

1. Pueden usar asistentes de IA como chat GPT para acelerar el desarrollo.
2. Se evaluará el **buen análisis y desarrollo de computación paralela.**
3. **Documente** las partes relevantes de su código.
4. Haga un repositorio github con su programa completo para descargar y ejecutar.
5. **Un informe completo bien hecho, sus gráficos, análisis y conclusiones del benchmark forman una parte importante de su nota.**

Grupos, entregables y fecha de entrega

- **[GRUPOS]** Pueden trabajar en grupos de hasta 4 personas.
- **[ENTREGABLES]** Archivo ***.zip** con código fuente + informe PDF de máximo 4 páginas.
- **[PLAZO]** **07 de Diciembre 2024, todo el día.**