Trabajo Práctico 1: Optimización secuencial

Computación paralela

Marzorati Denise
M-6219/7

20 de abril de 2018

Docentes de la materia

Nicolás Wolovick
Carlos Bederián

Características del hardware y del software

- CPU: Intel Core i7-3632QM @ 2.20GHz.
- Memoria: 8GB, DDR3, 2 canales.
- L1 cache: 256KB
- L2 cache: 1MB
- L3 cache: 3MB
- Compilador: GCC 7.1.0.
- Sistema operativo: Ubuntu 16.04.4, x86_64.

Metodología de trabajo

Para comenzar a atacar cada problema lo primero que se hará es realizar algo de profiling utilizando la herramienta perf, para poder identificar a dónde debe dirigirse el mayor esfuerzo en optimizar. Se hará un breve resumen con las optimizaciones realizadas que dieron resultados significativos tras repetir varias veces la ejecución del programa.

IntegralImage

Para obtener la versión más rápida posible se ha compilado el código con:
gcc integralimage.c -o integral -O2 -Wall -Wextra -std=c99 -msse4 -funroll-loops -lm -lgomp.

Al hacer profiling se pudo observar que la función rand() consumía gran parte del tiempo de computación, por lo que se buscó versiones más rápidas de la función, encontrando así la presentada en el código como fastRandSSE4() que es más veloz ya que hace uso de las instrucciones SSE4, generando cuatro números aleatorios por vez.

Otra mejora fue la de hacer un poco de unrolling en el loop donde se realizan todos los cálculos para la generación de la imagen.

Por otro lado, se intentó hacer algo de loop tiling, alignment y aliasing; dichas técnicas no aportaron mejoras al rendimiento del programa. Todos los intentos de mejoras algorítmicas fueron fallidos, incluso utilizando librerías eficientes.

Para FRAMES = 300 los resultados han sido los siguientes:

- Avg IPC: 2.61
- Avg percentage cache misses: 16.16%
- Avg time: 0.991633s
- Avg percentage stalled cycles, frontend: 18.61%

Para FRAMES = 3000 los resultados han sido los siguientes:

- Avg IPC: 2.69
- Avg percentage cache misses: 12.02%
- Avg time: 9.420287s
- Avg percentage stalled cycles, frontend: 17.61%

Para FRAMES = 30000 los resultados han sido los siguientes:

- Avg IPC: 2.68
- Avg percentage cache misses: 12.61%
- Avg time: 95.767781s
- Avg percentage stalled cycles, frontend: 17.89%

Con las optimizaciones realizadas se ha podido observar que ahora el tiempo que el programa tiene de ejecución crece linealmente con respecto al tamaño del mismo, en este caso dado por el valor FRAMES.