Arquitectura de Computadores

2º Curso Grao Enx. Informática

Práctica 2

Programación Multinúcleo y extensiones SIMD

Objetivos: programar un algoritmo simple con matrices en punto flotante, utilizando diferentes grados de optimización (utilización de varios núcleos, utilización de extensiones vectoriales SIMD, estrategias para reducir los fallos caché, etc.) y tomar medidas de rendimiento combinando diferentes optimizaciones y tamaños de problema.

Equipos: grupos de prácticas de dos personas.

Sistema para la toma de datos: FinisTerrae III del CESGA.

Tiempo de trabajo presencial: 5 sesiones.

Plazo de entrega: Para cada equipo **10 días** después de la última sesión de prácticas correspondiente. Comprobar la fecha límite en la tarea correspondiente del campus virtual.

Forma de entrega y formato: Electrónico. Se creará una tarea en el campus virtual para entregarlo. Se hará una única entrega por equipo consistente en una carpeta comprimida en formato ZIP y nombrada como Apellido1Apellido2_Apellido1Apellido2.zip con el siguiente contenido:

- Archivo PDF con la memoria de la práctica. (**máximo 8 páginas en total**).
- Código fuente completo de los diferentes apartados en una única carpeta nombrada "ACP2".

Valoración: 50% de la nota de prácticas (hasta 2.5 puntos sobre 10 en la nota total).

Objetivo: Hacer diferentes programas en C que realicen la computación del siguiente pseudocódigo de partida:

Entradas:

a[N][8], **b**[8][N], **c**[8]: matrices y vector que almacenan valores aleatorios de tipo double.

ind[N]: vector desordenado aleatoriamente que contiene índices de fila/columna sin que se repitan

Salida:

f: variable de salida tipo double

Computación:

```
d[N][N]=0; // inicialización de todas las componentes de d a cero;
 for (i=0; i<N; i++) {
    for(j=0; j<N; j++) {
       for (k=0; k<8; k++) {
         d[i][j] += 2 * a[i][k] * (b[k][j]- c[k]);
       }
     }
f=0;
for (i=0; i<N; i++) {
    e[i]= d[ind[i]][ind[i]]/2;
    f+=e[i];}
Imprimir el valor de f
```

NOTA: Para facilitar la comparativa entre las diferentes versiones de la práctica **debe computarse obligatoriamente el vector d** en su totalidad en todas las versiones.

Las diferentes versiones son las siguientes:

i) Programa secuencial base.
ii) Programa secuencial con optimizaciones basadas en el uso de memoria caché.
iii) Programa optimizado paralelizado utilizando paralelismo a nivel de datos (procesamiento vectorial SIMD).
iv) Programa optimizado paralelizado utilizando programación paralela en memoria compartida (OpenMP).

Detalles sobre las diferentes versiones:

i) **Programa secuencial base**

Se trata de implementar en C directamente el pseudocódigo anterior.

ii) **Programa secuencial con optimizaciones basadas en el uso de memoria caché:**

El objetivo de este apartado es realizar optimizaciones sobre el código del apartado anterior de modo que se obtenga al final el mismo vector resultado pero que se reduzca el tiempo de ejecución.

Para ello, implementar las siguientes mejoras:

1. Uso eficiente de registros para almacenamiento temporal de datos.
2. Reducción del número total de instrucciones a ejecutar.
3. Fusión y/o intercambio de lazos.
4. Desenrollamiento de lazos.
5. Realización de operaciones por bloques
6. Búsqueda de la mejor combinación de las optimizaciones anteriores.

iii) **Programa optimizado paralelizado utilizando paralelismo a nivel de datos (procesamiento vectorial SIMD).**

Realizar una implementación alternativa a la del apartado anterior, utilizando las extensiones AVX512 para conseguir paralelismo a nivel de datos.

iv) **Programa optimizado paralelizado utilizando programación paralela en memoria compartida (OpenMP).**

Por último, realizar otra implementación alternativa a la del apartado ii), utilizando paralelismo mediante OpenMP. Deben estudiarse los efectos de los siguientes factores:

1. Variación del número de hilos utilizados.
2. Empleo de distintos mecanismos de “scheduling”.
3. Uso de la cláusula “Collapse”.

En este apartado **no** está permitido ni el uso de variables tipo “reduction” ni el de extensiones vectoriales.

Experimentación:

Para la realización de los experimentos se proporciona un script que se encargará de la compilación y ejecución del código. El programa debe aceptar únicamente 2 parámetros como argumentos de entrada, correspondientes a los valores de N (tamaño) y C (número de cores empleado).

Resulta de especial interés para la comprensión de los experimentos comprobar en el manual del compilador el efecto de compilar los códigos con los distintos niveles de optimización automática.

Es requisito indispensable que el programa entregado funcione con este script para la corrección de la práctica.

Para el uso de extensiones vectoriales debe incluirse en el código fuente la cabecera "`#include <immintrin.h>`". Para el uso de OpenMP debe incluirse la cabecera "`#include <omp.h>`".

Hacer experimentos considerando que el número N de filas y columnas de la matriz toma los valores siguientes: **N=500, 750, 1000, 1500, 2000, 2500, 3000, 3500.** En todos los casos hacer reserva de **memoria dinámica** de las matrices y vectores e **inicializarlos** con valores aleatorios en un intervalo prefijado. **Medir el número de ciclos** de la parte del programa en que se hace la computación indicada en el pseudocódigo. **Asegurarse de que el resultado final en todas las versiones es el mismo, es decir, que todas las versiones realizan la computación correctamente.** Cuando se utilicen OpenMP o extensiones AVX512 se debe incluir en la medida de tiempo todo lo que implique una sobrecarga respecto del programa secuencial optimizado. Para cada caso, tomar al menos 15 medidas, y seleccionar la **mediana** de estos valores como valor final de medida de tiempo de ejecución.

Memoria de la práctica:

Formato:

Será un documento en PDF siguiendo la misma plantilla que se usó para la práctica 1 (título, autores, resumen, introducción, descripción de los experimentos, resultados y su análisis, conclusiones y bibliografía). Utilizar el mismo tamaño de letra y doble columna para la edición del texto.

Contenido de la memoria:

Es un informe técnico. Hay que explicar lo más relevante de los códigos asociados a cada apartado incluyendo un pseudocódigo, incluir gráficas de resultados e interpretación de dichos resultados. La memoria debe incluir las gráficas que se consideren oportunas para sacar conclusiones de los resultados obtenidos.

Es de especial interés representar la ganancia en velocidad (también llamada aceleración o *speedup*) de la versión secuencial optimizada con respecto a la versión inicial compilada con -O0, la ganancia en velocidad de los códigos de los apartados iii) y iv) con respecto a la versión secuencial optimizada, y finalmente las versiones desarrolladas en los apartados ii), iii) y iv) respecto de la versión inicial compilada con -O3).

En el caso del apartado iv), además de las gráficas que se consideren adecuadas, es imprescindible representar en una única gráfica la ganancia en velocidad conseguida para los diferentes números de hilos variando el valor de N y una gráfica separada mostrando el comportamiento para el valor más grande de N.

Criterios de evaluación:

Cumplir las especificaciones del enunciado de la práctica buscando siempre obtener la mínima latencia (tiempo de ejecución) del programa en las diferentes implementaciones, rigurosidad en el planteamiento e interpretación de resultados, exploración de alternativas, estudio autónomo, presentación de resultados (calidad de la memoria) y entrega en el plazo marcado.