**TRABAJO OPENMP – PRÁCTICA 4**

**ORDENACIÓN DE VECTORES**

**ARQUITECTURA DE COMPUTADORES**

**CURSO ACADÉMICO 2020/2021**

**Sergio Jiménez Roncero** (Laboratorio B1)

**Javier Álvarez Páramo** (Laboratorio B1)

El procesador que se ha usado para la toma de tiempos es un Intel Core i7-8750H, que comprende entre 2,2 y 4,1 GHz de frecuencia, disponiendo a su vez de 6 núcleos y 12 hilos.

Para que los tamaños creciesen de manera uniforme, y con ello se pudiese observar fielmente de qué forma crecen los tiempos, se ha utilizado una función matemática de una recta lineal, de tal forma que el valor de tamaño más pequeño fuese 10, y el valor de tiempo del tamaño más grande fuese al menos 1 minuto en algún algoritmo (algoritmo C):

**Apartado 1. Tarea 1: Tiempos de los métodos secuenciales**

Los valores resultantes han sido los que se muestran a continuación. La columna de la izquierda de cada algoritmo representa todos los valores de tiempo, mientras que la de la derecha representa la media de los 3 valores intermedios.



Datos en bruto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tamaño vector** | **Sec\_A** | **Sec\_B** | **Sec\_C** | **Sec\_D** |
| 10 | 0.002466667 | 0.0014 | 0.001733333 | 0.002233333 |
| 40010 | 597.5971 | 902.7557667 | 4132.993233 | 3157.0783 |
| 80010 | 2409.0845 | 3706.2063 | 17339.1465 | 12900.74187 |
| 120010 | 5460.799633 | 8388.892567 | 39430.47657 | 29290.97073 |
| 160010 | 9655.7741 | 15351.85873 | 69662.02607 | 52080.4518 |

Media de los tiempos excluyendo el menor y mayor tiempo



Análisis de los resultados obtenidos

Llevando a cabo un análisis de los resultados obtenidos, se puede observar en las distintas gráficas tamaño-tiempo que el crecimiento de los tiempos (eje Y), en función del crecimiento del tamaño (eje X), creemos que comprende una función de tipo parabólica o exponencial , de tal forma que las pendientes de los segmentos formados entre el resultado de una muestra y el resultado de la siguiente muestra crece cada vez más de manera general, tal como se puede observar en las gráficas.

Aún así, para tener una mayor certeza de lo afirmado en el anterior párrafo, hemos calculado las pendientes de cada segmento, y, la diferencia y relaciones entre cada pendiente y su anterior, para ver cómo se iban desarrollando.

Observamos que, de manera general, las pendientes crecen cada vez más, con una diferencia entre ellas oscilante para cada algoritmo.

La función como tal no es derivable porque no contempla todos los números reales, sino que va dando ciertos saltos entre las distintas muestras de un mismo algoritmo, y se producen cambios de pendientes instantáneos, bruscos (por eso no es derivable). Si a esto le añadimos que se pueden producir márgenes de error en las mediciones, se nos dificulta obtener la función matemática representativa, pero sí que creemos afirmar que es generalmente creciente, salvo los errores producidos. Apostaríamos por que se trate de una función parabólica puesto que la función de las pendientes (derivada) se sitúa en torno a una recta de Pearson, y porque la integral de una recta (derivada) es una parábola (función primitiva).

Podemos afirmar por tanto que, en la mayoría de las ocasiones, la función del tiempo crece cada vez más, aunque no con una tasa de crecimiento (derivada) exactamente constante o lineal, sino oscilante, pero creciente y decreciente en torno a una recta de Pearson.

**Resultados particulares de los métodos secuenciales:**

En algunos métodos de ordenación observamos que la tasa de crecimiento es mayor que en otros.

En el caso de los algoritmos secuenciales, observamos que el método que tarda más tiempo es el algoritmo C, mientras que el que tarda menos es el A, seguido del B:

**Apartado 2. Tarea 3: ¿Mejora de tiempos al paralelizar?**



Datos en bruto

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tamaño vector** | **Par\_A** | **Par\_B** | **Par\_C** | **Par\_D** | **Par\_Dm** |
| 10 | 0.06433333 | 0.0046 | 0.0055 | 0.03816667 | 0.02186667 |
| 40010 | 3551.68003 | 921.770967 | 4075.1758 | 838.2965 | 1072.52387 |
| 80010 | 14501.207 | 3840.27077 | 17468.222 | 3564.9611 | 4956.0295 |
| 120010 | 30867.6562 | 8545.181 | 39307.2969 | 8729.28367 | 10283.2376 |
| 160010 | 56339.3997 | 14738.5895 | 68586.3411 | 12781.0065 | 17283.3424 |

Media de los tiempos excluyendo el menor y el mayor tiempo



Análisis de los resultados obtenidos

Continuamos con todos los argumentos desarrollados en el anterior apartado, pero destacamos que, en el momento en el que decimos que la diferencia entre pendientes es generalmente creciente, indicamos que se trata en general en la mayoría de los casos, pero particularmente en el algoritmo D paralelo, existe un momento en el que no es así. La diferencia pasa a ser negativa, es decir, ha disminuido la pendiente con respecto a la anterior, para el caso del 4º segmento.

Hemos encontrado esa diferencia en el algoritmo D, cuya pendiente es menor (más plana), y, que probablemente el decrecimiento de la derivada se deba a errores producidos en las mediciones.

**Resultados particulares de los métodos paralelos:**

Lamentablemente, los resultados de los tiempos del algoritmo A paralelizado no han mejorado con respecto a los tiempos del algoritmo A secuencial. Empeoran considerablemente.

En el caso del algoritmo D paralelizado no mejorado, sí que mejora con respecto al D secuencial. El algoritmo D paralelizado mejorado con respecto al algoritmo D paralelizado no mejorado, empeora.

Creemos que los algoritmos B y C no se pueden paralelizar por los motivos señalados en los comentarios del código.

En este caso particular de tomar los algoritmos paralelos o secuenciales en el caso de no ser paralelizables, el algoritmo C secuencial/no paralelizable continúa siendo el que más tiempo tarda, seguido del A paralelo; mientras que el que menos es el algoritmo D paralelo seguido del B no paralelizable. Entre medias se encuentra el algoritmo D paralelo mejorado.

Se hace entrega en Ciudad Real, a martes, 1 de diciembre de 2020.

(Fecha límite de entrega para el Laboratorio B1: jueves, 3 de diciembre a las 19:59 horas)