

MEDIDAs empíricas de algoritmos



20 de febrero de 2018

Gema rico pozas

UO238096

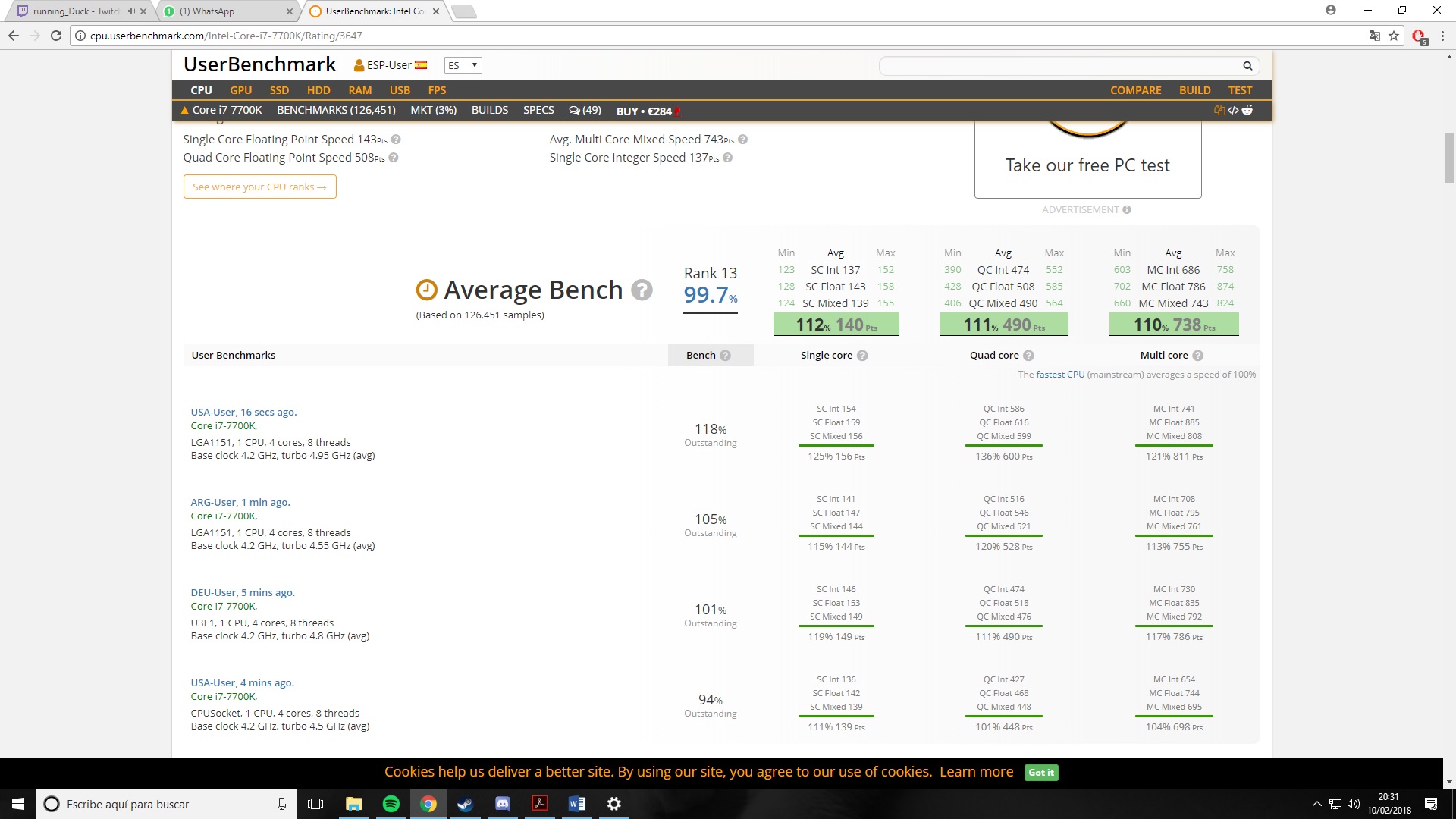
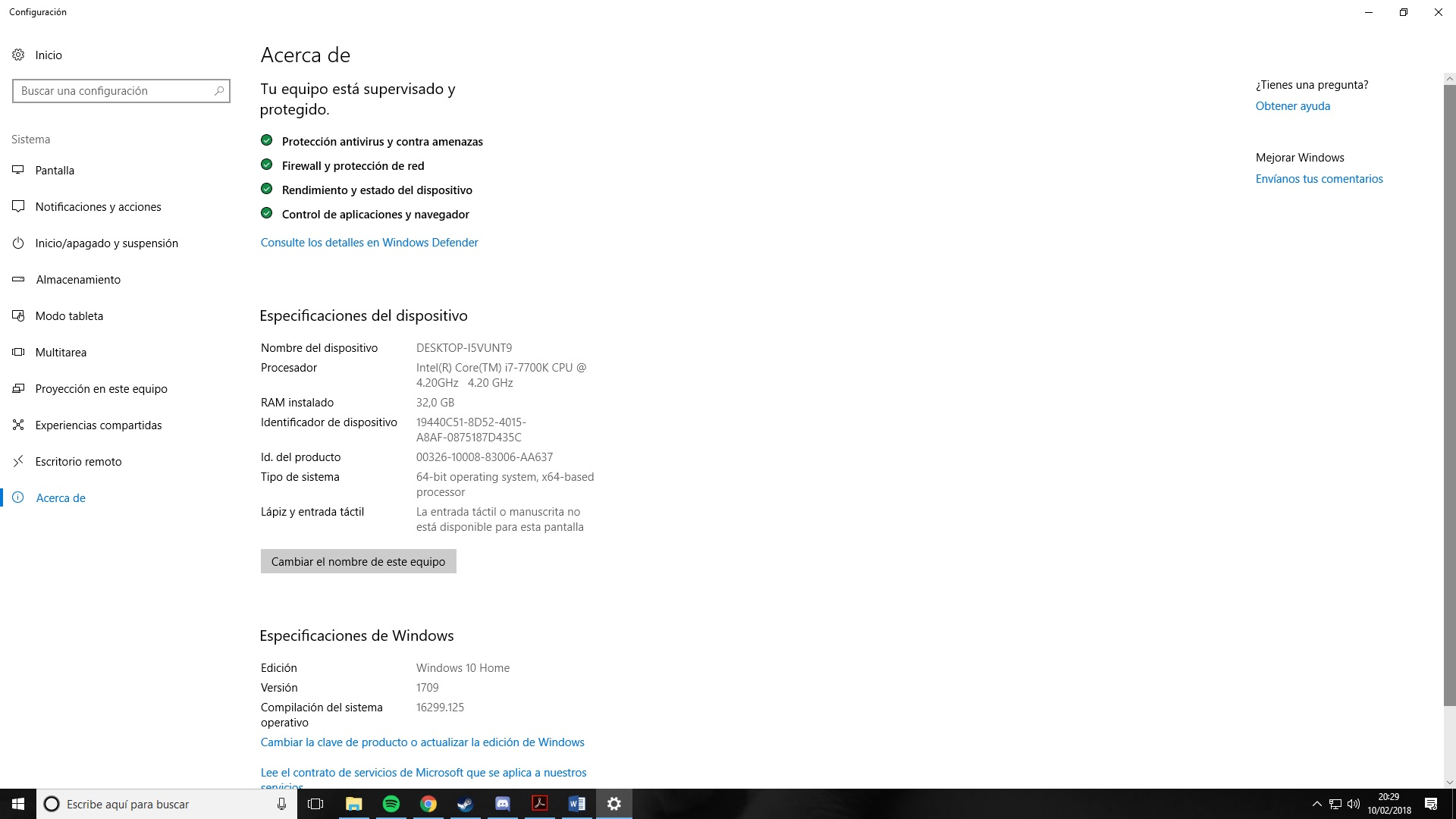
PRACTICA 1

TAREA 1: POTENCIA DE LAS CPUS

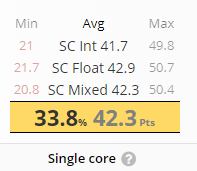
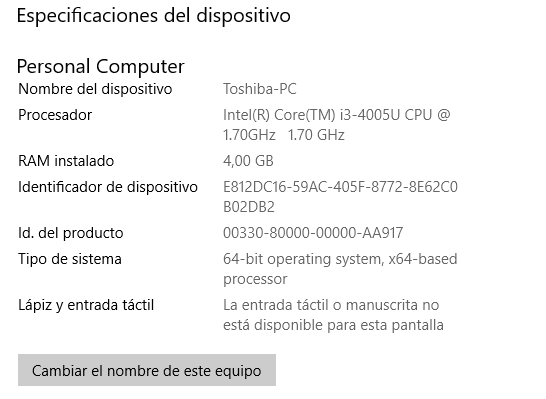
La primera actividad que vamos a realizar es la comparación en varios ordenadores el tiempo de ejecución de varios programas para ver la influencia de las CPU.

Para ello realizo las mediciones con mi ordenador de sobremesa y con mi ordenador portátil.

Ordenador de sobremesa:



Portátil:



Compilamos y ejecutamos el programa *Benchmarking1* y anotamos el tiempo que ha tardado en ejecutar en los dos pc.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | CPU | milisegundos | SC int (max) | Operaciones (aprox.) |
| 1 | IntelCore i7-7700K | 175 | 152 | 26600 |
| 2 | IntelCore i3-4005U | 798 | 49.8 | 39740 |

Conclusión

Viendo los resultados en milisegundos, ¿crees que podrías mezclar valores de diferentes CPUs en un mismo estudio analítico de los tiempos de ejecución de un algoritmo?

No se deben mezclar puesto la CPU influye de manera sustancial en la medidad de los mismos como se puede comprobar en los resultados anteriores.

TAREA 2: INFLUENCIA DEL SISTEMA OPERATIVO

En esta actividad analizaremos las políticas de administración de energía del procesador y la ejecución simultánea de varios procesos sobre la misma CPU puesto que gran parte de la responsabilidad de gestionar estos recursos recae en el sistema operativo.

Ejecutamos el programa anterior (benchmarking2) de dos maneras:

Ejecución secuencial: Ejecutamos el programa múltiples veces con el script run.cmd. sin realizar ninguna otra operación más realizando las mismas pruebas con diferentes planes de energía.

Ejecución paralela: Ejecutamos el programa cpuburn.exe. Este programa consume el 100% de la paralelamente ejecutamos el programa Benchmarking1 múltiples veces (run.cmd).

Conclusiones

¿Qué plan de energía crees que es el más adecuado para realizar mediciones?

Alto Rendimiento

Si tuvieses que realizar la medición de un experimento muy largo, ¿podrías utilizar el ordenador para por ejemplo ver un vídeo de YouTube?

No porque consume recursos del ordenador y altera notablemente el tiempo de medición del experimento.

¿Crees conveniente realizar varias mediciones simultáneamente en el mismo ordenador?

No, porque estaríamos alterando ambas mediciones.

TRABAJO PEDIDO: MEDICIÓN DE VARIOS ALGORITMOS

Realizamos la medición de varios algoritmos proporcionados en clase con diferentes cargas de trabajo para comparar que la carga de trabajo y complejidad del algoritmo influye en los tiempos de ejecución.

Para la medida de estos tiempos utilizaremos el ordenador de sobremesa citado anteriormente.

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | Tiempos “rellena” | Tiempos “suma” | Tiempos “maximo” |
| 10 | 0,7 | 0,008 | 0,0201 |
| 50 | 2 | 0,0312 | 0,0403 |
| 250 | 6 | 0,2247 | 0,1647 |
| 1250 | 28 | 0,8953 | 0,7349 |
| 6250 | 139 | 5 | 3 |
| 31250 | 696 | 24 | 18 |
| 156250 | 3458 | 145 | 90 |
| 781250 | 17486 | 840 | 782 |
| 3906250 | 74548 | 3320 | 3961 |
| 19531250 | 394614 | 16988 | 20040 |
| 97656250 | 2215486 | 78201 | 102458 |

🡪¿cumplen los valores obtenidos con lo esperado?

Si puesto que, a pesar que todos los algoritmos tienen una complejidad O(n), la función rellena ocupa muchos más recursos que suma y máximo ya que va generando numeros aleatorios para cada elemento del array.

|  |  |
| --- | --- |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | Tiempos “rellena” | Tiempos “diagonal 1” | Tiempos “diagonal 2” |
| 3 | 0,2 | 0,009 | 0,003 |
| 6 | 0,4 | 0,05 | 0,006 |
| 12 | 2 | 0,1 | 0,01 |
| 24 | 7 | 0,5 | 0,02 |
| 48 | 29 | 2 | 0,04 |
| 96 | 121 | 7 | 0,08 |
| 192 | 454 | 23 | 0,2 |
| 384 | 2075 | 79 | 0,7 |
| 768 | 7352 | 310 | 1 |

PRACTICA 2

TRABAJO PEDIDO: MEDICION DEL TIEMPO EN ALGORITMOS ITERATIVOS / CONSTANTE DE IMPLEMENTACION DE DOS ALGORITMOS

Todas las mediciones se realizan utilizando el ordenador de sobremesa citado anteriormente.

Tabla 1. Dos algoritmos con misma complejidad; bucle2 y bucle3 (ambos O(n2)) y así obtener la constante bucle2/bucle3.

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | Tiempos “bucle2” | Tiempos “bucle3” | Tiempos “bucle2/3” |
| 8 | 1,16 | 1,17 | 0,991452991 |
| 16 | 4,48 | 4,56 | 0,98245614 |
| 32 | 18,02 | 17,89 | 1,007266629 |
| 64 | 0,075 | 0,082 | 0,914634146 |
| 128 | 296 | 281 | 1,053380783 |
| 256 | 1139 | 1156 | 0,985294118 |
| 512 | 4432 | 4545 | 0,975137514 |
| 1024 | 21752 | 18426 | 1,180505807 |
| 2048 | 110569 | 85475 | 1,293582919 |
| 4096 | 357196 | 290001 | 1,231706098 |

Los resultados son los esperados. Como podemos observar en los gráficos, bucle 2 y 3 presentan complejidad cuadrática. La constante bucle2/3 resultante de dividir tiempos de bucle2 / tiempos de bucle3. Podemos observar que bucle 3 es mas rápido con n a partir de 1024.

Tabla 2. Dos algoritmos con distinta complejidad; bucle1 y bucle2 (uno O(n logn) y otro O(n2)) y así obtener la constante bucle1/bucle2.

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | Tiempos “bucle1” | Tiempos “bucle2” | Tiempos “bucle1/2” |
| 8 | 1,2 | 1,1 | 1,091 |
| 16 | 4,8 | 4,5 | 1,067 |
| 32 | 17,9 | 17,7 | 1,011 |
| 64 | 69 | 70 | 0,986 |
| 128 | 304 | 321 | 0,947 |
| 256 | 1200 | 1186 | 1,012 |
| 512 | 4651 | 4432 | 1,049 |
| 1024 | 18650 | 22042 | 0,846 |
| 2048 | 76357 | 102950 | 0,742 |
| 4096 | 301469 | 331200 | 0,910 |

Se puede apreciar en los gráficos que a medida que la carga de trabajo aumenta el algoritmo con complejidad O(nlogn ) es decir el bucle 1 es más rápido que el algoritmo con complejidad cuadrática, bucle2, por tanto los datos se corresponden con lo esperado.

Tabla 3. Complejidad del resto de los algoritmos;Implementamos bucle4 y bucle5, simulando algoritmos iterativos con una complejidad O(n4) y O(n3logn) respectivamente.

|  |  |  |
| --- | --- | --- |
| Carga de trabajo (n) | Tiempos “bucle4” | Tiempos “bucle5” |
| 8 | 50 | 35 |
| 16 | 1147 | 400 |
| 32 | 24569 | 4562 |
| 64 | 300000 | 41865 |
| 128 | 4786000 | 301040 |
| Complejidad | O(n4) | O(n3logn) |

Además, se debe medir los tiempos de la última clase proporcionada incognita y deducir su complejidad.

|  |  |
| --- | --- |
| Carga de trabajo (n) | Tiempos “incógnita” |
| 8 | 0,6 |
| 16 | 0,33 |
| 32 | 24569 |
| 64 | 300000 |
| 128 | 4786000 |
| Complejidad | O(n3) |

La clase incógnita tiene una complejidad O(n3) puesto que tiene tres bucles iterativos anidados.