Memoria

Prácticas

Algoritmia Básica

Práctica 2

Autor:

Hugo Mateo Trejo, 816678

Paula Oliván Usieto, 771938

Algoritmo:

Debido al gran coste que tenía el algoritmo de fuerza bruta se decidió mirar la norma que seguían las bolas al caer por los nodos para no tener que lanzar todas las bolas anteriores a una en concreto para saber a qué hoja caerá. Es fácil darse cuenta de que cada dos bolas que pasan por un nodo la siguiente bola en pasar por esa zona pasará por el mismo camino que la primera bola lanzada.

Por ello por ejemplo en el nodo raíz las bolas impares siempre caen a su subárbol izquierdo y las bolas pares caerán por tanto en el subárbol derecho. Viendo que esto es predecible matemáticamente solo es necesario transportar este algoritmo al resto de nodos del árbol. Quedando la siguiente fórmula: $(i-1)\%\,2^{\,p}$ siendo i la bola a colocar y p el nivel donde nos encontramos.

Pruebas realizadas:

Para garantizar la corrección se han comparado los caminos resultantes de las tres implementaciones. Dado que el algoritmo de FB genera el árbol completo, a nivel algorítmico no puede tener errores. A nivel de código, se ha depurado comparándolo con resoluciones pequeñas hechas a mano. En todas las pruebas hechas todos los algoritmos han devuelto el mismo resultado que el de fuerza bruta.

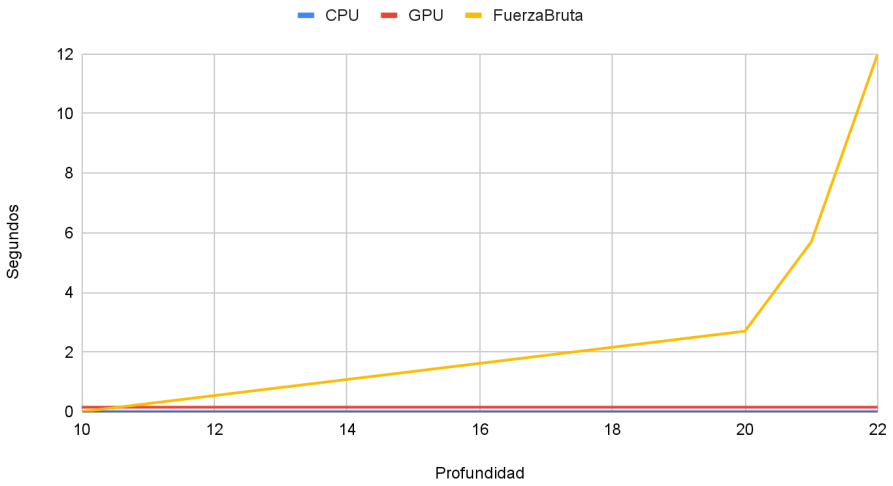
```
Lanzamos el programa para probar los datos de pruebas.txt
Tiempo de ejecucion del algoritmo en CPU: 4.57763671875e-05
Tiempo de ejecucion del metodo de fuerza bruta: 0.0006742477416992188
Los caminos son iguales
Camino: [1, 1, 1]
Tiempo de ejecucion del algoritmo en CPU: 8.177757263183594e-05
Tiempo de ejecucion del metodo de fuerza bruta: 0.07302236557006836
Los caminos son iguales
Camino: [1, 1, 0, 0, 1, 1, 1, 1, 1]
Tiempo de ejecucion del algoritmo en CPU: 6.842613220214844e-05
Tiempo de ejecucion del metodo de fuerza bruta: 0.018614768981933594
Los caminos son iguales
Camino: [1, 1, 1, 1, 1, 1, 1]
Tiempo de ejecucion del algoritmo en CPU: 4.9114227294921875e-05
Tiempo de ejecucion del metodo de fuerza bruta: 0.0008931159973144531
Los caminos son iguales
Camino: [1, 0]
Tiempo de ejecucion del algoritmo en CPU: 5.316734313964844e-05
Tiempo de ejecucion del metodo de fuerza bruta: 0.0014879703521728516
Los caminos son iguales
Camino: [1, 1, 1, 1]
Tiempo de ejecucion del algoritmo en CPU: 6.222724914550781e-05
Tiempo de ejecucion del metodo de fuerza bruta: 0.0070879459381103516
```

PosicionBolasLanzadas	FuerzaBruta	DivideYVenceras
500	0.07302236557006836	-0.00014090538024902344
128	0.018614768981933594	-0.0001380443572998047
2	0.0008931159973144531	-0.00014162063598632812
16	0.0014879703521728516	-0.00013899803161621094
128	0.0070879459381103516	-0.00013828277587890625

Análisis de los resultados:

Como podemos observar en las pruebas el algoritmo de fuerza bruta es incapaz de generar árboles que superen la profundidad 22 ya que se genera un árbol que ocupa demasiado espacio en memoria. Por el contrario el algoritmo de divide y vencerás no tiene problema en calcular donde caerá una bola i a profundidades de 1.6e+10, todo esto debido a que en su caso la limitación existente es temporal no de memoria.

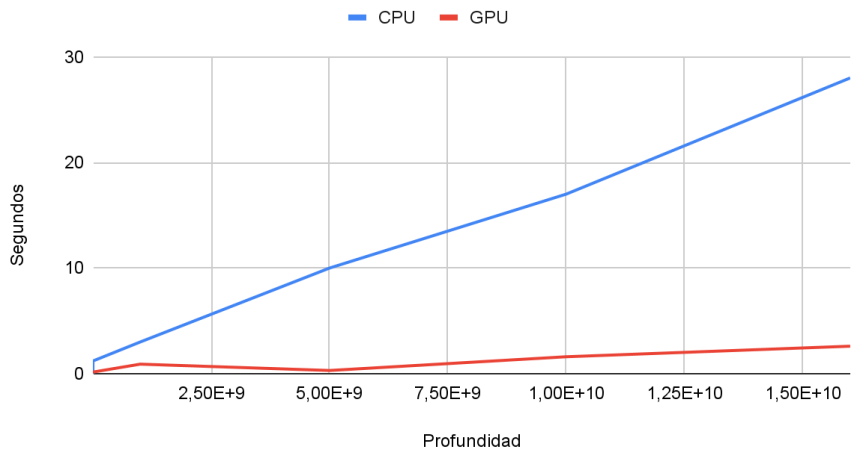
Aún así ninguno de los dos algoritmos puede hacer frente al coste de la realización del algoritmo en GPU ya que con las pruebas que en CPU tardan alrededor de medio minuto en GPU se mantienen aún por debajo del segundo. Debido a que no se podrán lanzar estas pruebas en Hendrix ni eran requisito de la asignatura se dejan como anécdota y demostración de la potencia que posee este hardware para este tipo de cálculos matemáticos.



Como podemos observar en el gráfico tanto el algoritmo de divide y vencerás en CPU y GPU para los árboles de profundidad 22 no ha sufrido prácticamente cambios en la duración para obtener resultados. No sucede lo mismo al aplicar el algoritmo de fuerza bruta donde podemos ver que el tiempo ha incrementado muy bruscamente al llegar a la zona de profundidad 22, a lo cual hay que sumarle que no se han podido realizar más pruebas de comparación de los tres algoritmos ya que como se ha nombrado anteriormente se llega al límite de la memoria.

Se aprecia claramente que el algoritmo de fuerza bruta crece en 2^p. El algoritmo divide y vencerás es lineal en p, como se puede apreciar en los tiempos. Cada vez que la profundidad se multiplica por 10, el tiempo también lo hace. La implementación de CPU es más rápida en profundidades pequeñas, debido al coste constante añadido de la comunicación con la GPU (el pci express es lento). Para p grandes la GPU supera a la CPU, aunque el valor exacto es difícil de calcular debido a los costes añadidos de la compilación de numba (1.4s cte).

Gracias a la compilación el coste aumenta linealmente, pero con una constante menor a 1. Concretamente, al aumentar x10 la profundidad el tiempo aumenta x5.8 en media. En la GPU, aumentar x10 la p aumenta el tiempo en x5.6



Una vez eliminado el algoritmo de fuerza bruta y probando los límites de nuestro algoritmo de divide y vencerás en CPU y GPU. Se puede diferenciar como al igual que pasaba anteriormente cuando llegábamos a los límites del algoritmo de fuerza bruta el tiempo en el algoritmo al ser ejecutado en CPU aumenta constantemente su tiempo de ejecución linealmente mientras que la GPU mantiene un tiempo bajo durante la totalidad de las ejecuciones.

A pesar de esto, hemos llegado a una profundidad de 1,5e+10 lo cual es una profundidad muy alta y no hemos llegado a los 30 segundos de ejecución, recordemos que con una cantidad infinitamente menor de profundidad el algoritmo de fuerza bruta tardaba la mitad en ejecutarse. Esto se ve más claro en la tabla adjuntada en la parte inferior que muestra todos los tiempos según profundidad.

Profundidad	CPU	GPU	FuerzaBruta
10	0,00001	0,15	0,0005
20	0,000012	0,15	2,7
21	0,000013	0,15	5,7
22	0,000013	0,15	12
1000	0,0001	0,15	No probado
10000	0,001	0,15	No probado
100000	0,03	0,15	No probado
1000000	1,2	0,15	No probado
1000000000	3	0,9	No probado
5000000000	10	0,3	No probado
10000000000	17	1,6	No probado
16000000000	28	2,6	No probado

Podemos concluir por tanto diciendo que todas las implementaciones del algoritmo divide y vencerás escalan claramente de forma lineal. La compilación reduce el factor de escala en 1.6, y la GPU reduce el tiempo base entre 10, pero todas escalan linealmente.