Trabajo práctico Monitores – Programación Concurrente

Autores:

Tobías Torres – tobiastorres777@gmail.com – legajo: 36046

Matias Cardozo – matiascardozo2795@gmail.com – legajo: 38250

Introducción:

El siguiente trabajo práctico trata de simular el funcionamiento del algoritmo “Proof of Work” de la red Bitcoin, con una dificultad de procesamiento acotada a modo de estudio de la misma.

El objetivo principal se centra en recrear la funcionalidad del ya mencionado algoritmo “Proof of work”. Que dada una dificultad (medida en bytes en este caso), y una cadena de bytes, tras aplicarle el algoritmo SHA-256 encontrar la cadena que tenga los primeros *n* bits en cero (donde *n* es la dificultad previamente dada). A dicha cadena resultante la llamaremos “*nonce”*.

Diseño del código:

Creamos 6 clases las cuales son:

- **Main**, en esta clase tenemos el punto de entrada al programa, en dicha clase tenemos instanciados el buffer parametrizado con una capacidad de 2, y un ThreadPool .
- **Buffer**, es un monitor con una estructura de cola First In First Out que utiliza métodos synchronized y, dependiendo la capacidad pasada por parámetros, bloquea a los otros powWorkers cuando el buffer está lleno.

- **PowWorker**, en esta clase tenemos instanciados el Buffer y WorkUnity, extiende de Thread. Se encarga de realizar la búsqueda del *nonce.*
- **Range**, clase en la que se generan los rangos de inicio y fin, que luego cada powWorker utilizará.
- **ThreadPool**, se encarga de iniciar e instanciar la cantidad de PowWorkers pedidos por el usuario.
- **WorkUnity**, esta clase se encarga de instanciar texto, rango, dificultad, será la encargada de encontrar en nonce en el rango que tome para trabajar, a cada número de dicho rango lo convierte con el algoritmo “SHA-256” y checkea si el resultado es el esperado.

Evaluación:

Detalles del equipo de prueba

- Procesador: Intel i3 7100 3.9GHz
- Ram: 8gb 2400MHz DDR4 single channel
- Sistema operativo: Windows 10 Home 64bits

Tabla de resultados

Threads	Dificultad 2	Dificultad 3
1	**0.21s**	**0.59s**
2	**0.08s**	**0.71s**
4	**0.10s**	**1.01s**
6	**0.12s**	**1.2s**
8	**0.13s**	**1.8s**
10	**0.14s**	**2.2s**

Golden nonce: 4 minutos 54 segundos, utilizando 2 threads y dificultad 4.

Gráfico de tiempo de búsqueda de nonce con dificultad 2:

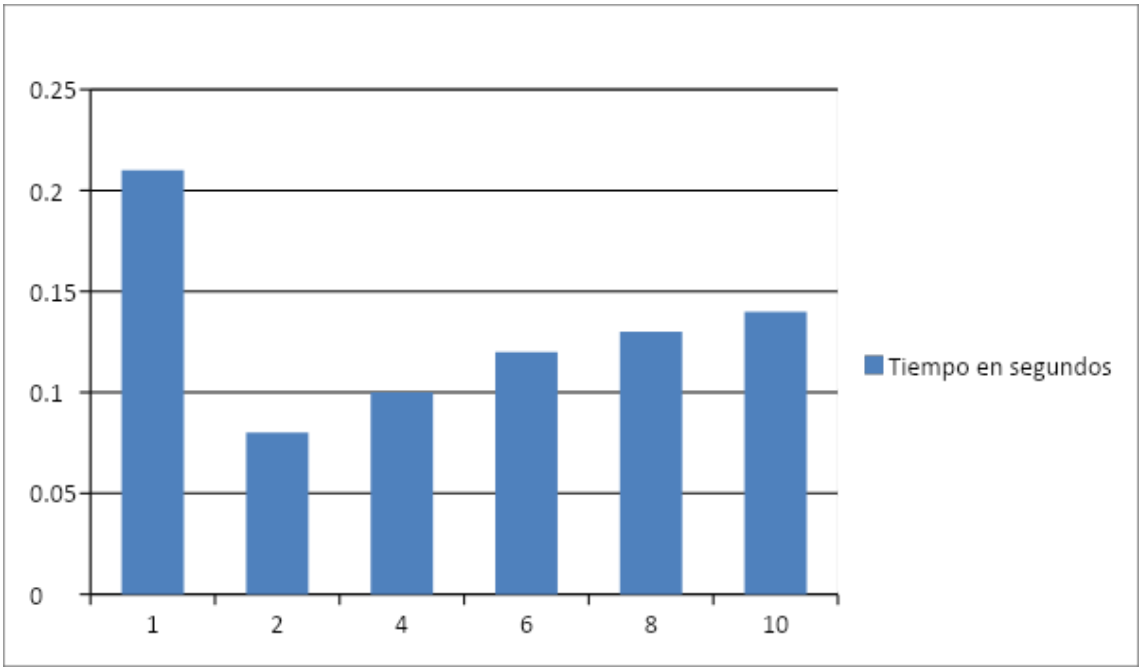


Gráfico de tiempo de búsqueda de nonce con dificultad 3:

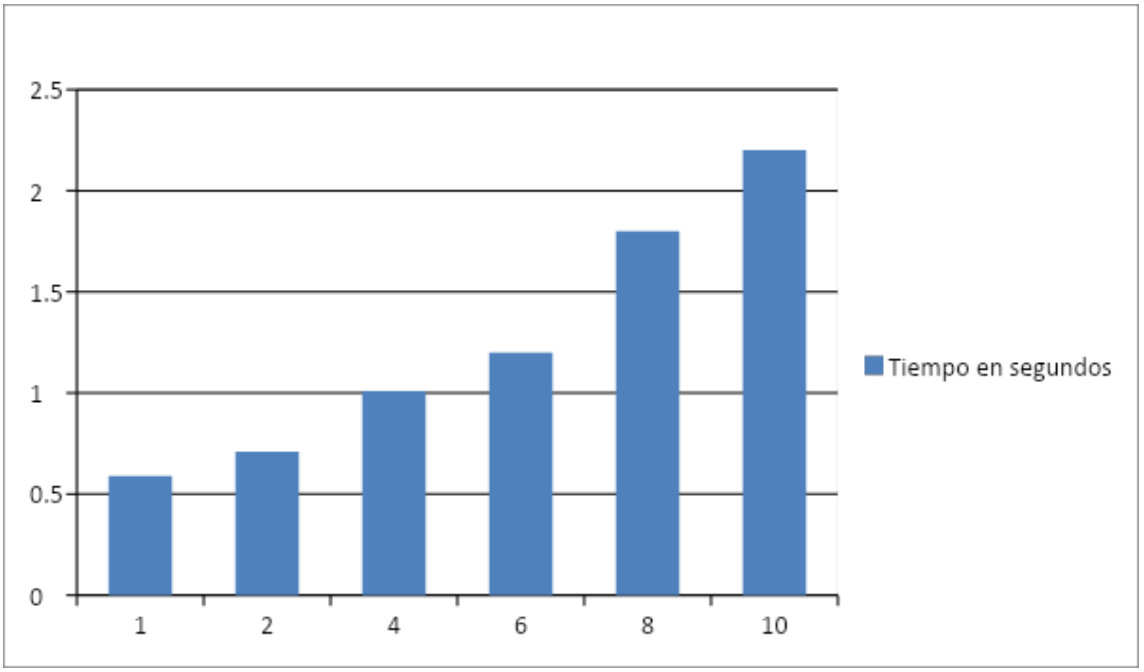
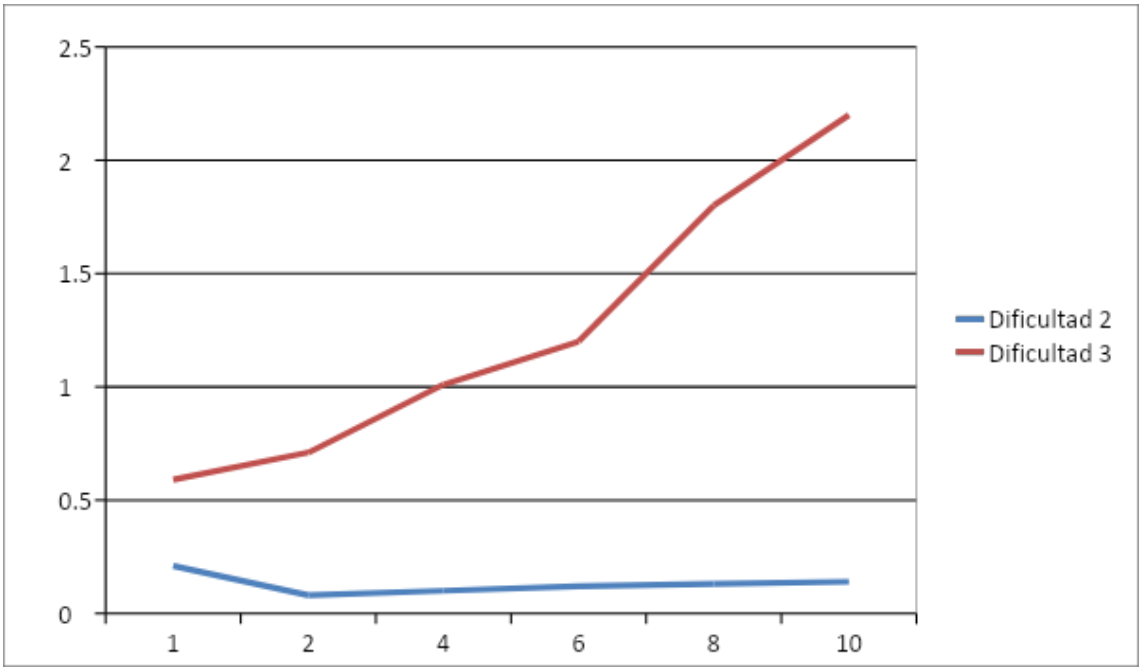


Gráfico superpuesto de ambas tablas



Análisis:

Tras haber realizado las pruebas correspondientes, se pudo observar que conforme se aumentaba el número de threads utilizados, aumentaba el tiempo en el que se encontraba el nonce. No solo se vieron reflejados estos aumentos de tiempo testeando con dificultad 2, sino que también con dificultad 3. Cabe destacar que dichos resultados corresponden al testeo con el hardware previamente detallado, se desconoce si utilizando un procesador con mayor frecuencia y threads mejoraría los resultados. La mayor eficiencia se pudo ver utilizando 2 threads, dicha diferencia es mínima de pocos milisegundos. Por lo cual podemos afirmar que tanto para 1 como para 2 threads puede tener un tiempo similar.

En cuanto al tiempo transcurrido hasta que se logró encontrar el “golden nonce” fue 4 minutos y 54 segundos, y se ejecutó con 2 threads y dificultad 4.