Sistemas Distribuidos I
(75.74)

Centralized Blockchain

TP1: Concurrencia y Comunicaciones

Docentes

- Pablo D. Roca
- Ezequiel Torres Feyuk
- Guido Albarello
- Ana Czarnitzki
- Cristian Raña

Brief summary Blockchain

- **Estructura de datos**: Lista enlazada donde cada bloque tiene un header y un payload. A cada nodo se le aplica una función de hash que identifica al bloque. Esta función de hash también toma como parámetro el hash del bloque previo (de aca proviene la integridad)
- **Minado**: Minar es la operación de agregar un bloque a la blockchain. Para ello un worker debe generar un bloque cuyo hash sea menor a cierto número. Mientras menor sea el número, más difícil es minar un bloque. El trabajo de cada minero es generar un bloque que cumpla con dicha condición, para ello se le concatena un nonce al bloque y se hashea el conjunto. En cada iteración el worker va incrementando el nonce hasta que se cumpla la condición

Requerimientos Funcionales

- Se solicita un sistema distribuido que brinde la funcionalidad de una *blockchain*
- Los usuarios deben ser capaces de almacenar un chunk de datos en la blockchain
- Los chunks que envían los usuarios se batchean en un bloque que será almacenado en la blockchain. Un bloque está compuesto a lo sumo de M chunks
- La máxima cantidad de chunks por bloque (M) es de 256 (especificado en el formato de bloque propuesto como **entries_amount**)
- El tamaño de cada chunk no puede superar los 65536 bytes

Requerimientos Funcionales

- Un hash es considerado válido en la blockchain si el hash **sha256** del bloque es menor a cierto número. Este número se calcula como: (2^256)/difficulty. La dificultad se almacena junto al bloque y es calculada dinámicamente.
- Cada 256 bloques minados se ajusta la dificultad según la ecuación => difficulty = prev_difficulty * (12s / (ELAPSED_TIME / 256))
- El sistema debe proveer la siguiente interfaz/contrato:
  - Almacenar un chunk de datos en la blockchain
  - Obtener un bloque de la blockchain
  - Obtener los bloques minados dado un intervalo de 1 minuto
  - Obtener cantidad de bloques minados de forma exitosa y errónea por cada miner

Requerimientos Funcionales

- Formato de bloque propuesto. Ejemplo de blockchain

```
header:
  - prev_hash: 32 bytes
  - nonce: 32 bytes
  - timestamp: 4 bytes (unix time)
  - entries_amount: 1 byte
  - difficulty: 32 bytes
entries:
  - entry_1: máximo 65356 bytes
  - entry_2
  - ...
  - entry_N

* prev_hash, nonce y difficulty deben ser de 32 bytes
* Tamaño de entries_amount y entry size definidos en el enunciado
```

Requerimientos No Funcionales

- Se espera una cantidad masiva de lecturas de los bloques de la blockchain y una gran cantidad de escrituras a la misma de distintos usuarios
- El sistema debe estar preparado para coordinar múltiples workers con el objetivo de aumentar las chances de minar un bloque
- El sistema debe estar optimizado para lecturas de bloques
- Por cuestiones de auditoría, la blockchain debe ser almacenada y accedida desde un único nodo aislado del resto del sistema y accedido a través de la red por un protocolo definido por el usuario
- Se espera que el sistema se encuentre **disponible** en todo momento, permitiendo que descarte bloques en caso de no poder procesarlos
- No está permitido almacenar todos los bloques en un único archivo

Normas de Trabajo

Se espera del alumno:

- Empleo del tiempo de consultas en clase para resolver dudas y clarificar el negocio del sistema a construir previo a su diseño
- Exposición y verificación en clase de la arquitectura propuesta previo a su implementación
- Empleo del grupo de correos para realizar consultas que no pudieran ser resueltas en clase
- Consideración de prácticas distribuidas según lo estudiado en clase para elaborar una arquitectura flexible, escalable y robusta
- Aprobación del cuerpo docente para el uso de cualquier librería
- Demo del sistema en funcionamiento previamente ensayada

Normas

- Fecha de entrega:
  - 11/05/2021
- Fecha de re-entrega:
  - 27/05/2021
- Formato de entrega:
  - Demostración del sistema utilizando Docker.
  - Entrega digital mediante correo personal incluyendo link al repositorio git, *tag* de la entrega e informe técnico que contenga:
    - Carátula
    - Diagrama de clases y detalle de las mismas
    - Diagrama de robustez o despliegue
    - Diagrama de actividades