Solana: Un Análisis de su Arquitectura y Escalabilidad en la Web3

Jaime Muñoz

Universidad de Antioquia

Jaime.munozq@udea.edu.co

Introducción

En la mayoría de las blockchains públicas, cada nodo utiliza su propio reloj local sin tener conocimiento de los relojes de otros participantes, lo que genera incertidumbre al aceptar o rechazar mensajes con base en sus marcas de tiempo. Solana propone una solución llamada Proof of History (PoH), que crea un registro de eventos con una verificación confiable del paso del tiempo. Esto permite a los nodos confiar en la secuencia temporal registrada en el ledger, lo que mejora el ordenamiento de mensajes y la sincronización en la red sin necesidad de confianza externa [1].

El Proof of History (PoH) es un mecanismo criptográfico utilizado en Solana para verificar el paso del tiempo entre dos eventos, sin depender de relojes externos. Funciona ejecutando una función criptográficamente segura, donde el resultado no puede predecirse a partir de la entrada, y debe ser completamente calculado para generar una salida válida.

Este proceso sigue los siguientes pasos:

- Ejecución secuencial: La función se ejecuta en secuencia en un solo núcleo de CPU, donde cada salida generada sirve como entrada para la siguiente operación. Esta cadena continua de cálculos crea una secuencia verificable.

- Registro periódico: Durante la ejecución, la función registra periódicamente su salida y el número de veces que ha sido llamada. Estos registros permiten revalidar la secuencia.

- Verificación paralela: Aunque la función se ejecuta en un solo núcleo, los resultados pueden verificarse en paralelo por otros nodos de la red. Cada segmento de la secuencia puede ser revisado de manera independiente por múltiples núcleos, lo que facilita la validación rápida.

- Marcas de tiempo: Los datos pueden añadirse a esta secuencia de tiempo mediante la inclusión de los datos o un hash dentro del estado de la función. Esto genera una marca de tiempo que garantiza que los datos fueron creados antes de que se generara el siguiente hash en la secuencia.

- Escalabilidad horizontal: El diseño de PoH soporta la escalabilidad horizontal. Múltiples generadores pueden sincronizarse entre sí mezclando sus estados dentro de las secuencias de los otros, permitiendo que diferentes nodos trabajen en paralelo.

Solana se ha diseñado para el uso generalizado y masivo al ser eficiente en términos de energía, extremadamente rápida y muy económica. Los creadores de Solana, como Anatoly Yakavenko, provienen del ámbito de redes de telefonía móvil, lo que les ha permitido enfocarse en la escalabilidad y la eficiencia. Estas características son fundamentales para que el público adopte y utilice la tecnología blockchain de manera más accesible [2].

A diferencia de otras blockchains que dependen de procesos de minería intensivos en energía y tiempo, Solana utiliza un mecanismo de validación llamado proof of stake, eliminando la necesidad de minería. Además, introduce la innovación del proof of history, que le permite validar transacciones de forma aún más rápida. Esto no solo reduce drásticamente el consumo energético, haciéndolo comparable al de unas pocas búsquedas en Google, sino que también reduce las tarifas de transacción a fracciones de centavo, en contraste con otras redes blockchain donde los costos pueden alcanzar cientos de dólares por transacción [8].

Así pues, Solana no solo ofrece una infraestructura altamente escalable y eficiente, sino que también redefine los límites de la tecnología blockchain al proponer soluciones innovadoras como Proof of History. Esta combinación de velocidad, bajo costo y sostenibilidad energética hace que Solana sea una de las blockchains más prometedoras para aplicaciones descentralizadas (dApps), finanzas descentralizadas (DeFi), y otros casos de uso que requieren un alto rendimiento. Su capacidad para soportar miles de transacciones por segundo (TPS) posiciona a Solana como un actor clave en la adopción masiva de blockchain, acercándonos a un futuro donde la tecnología descentralizada sea parte integral de la vida cotidiana.

¿Por qué solana?

Solana busca una solución distinta al "trilema de la blockchain" un concepto propuesto por Vitalik Buterin, el creador de Ethereum, que sugiere que es extremadamente difícil para una blockchain optimizar simultáneamente tres propiedades clave: **descentralización**, **seguridad** y **escalabilidad**. Esto lo logra a

través de su enfoque de consenso híbrido entre Proof of Stake (PoS) y Proof of History (PoH) [3].

PoH es clave en este enfoque porque elimina la necesidad de que los nodos de la red se comuniquen constantemente para sincronizarse y acordar el paso del tiempo. Esto no solo reduce drásticamente la latencia, sino que permite a Solana escalar sin los cuellos de botella típicos de otras blockchains. Mientras que otras redes requieren confirmaciones de múltiples nodos antes de validar un bloque, Solana puede mantener una red distribuida que alcanza velocidades superiores a 50,000 TPS gracias a PoH, al mismo tiempo que garantiza la integridad de la secuencia de eventos.

Este mecanismo también habilita a Solana para alcanzar tiempos de bloque significativamente más cortos (400 ms) en comparación con otros sistemas. De esta forma, Solana mantiene la seguridad y descentralización al tiempo que iguala la velocidad y capacidad de sistemas centralizados como los utilizados en transacciones bancarias o de tarjetas de crédito [4].

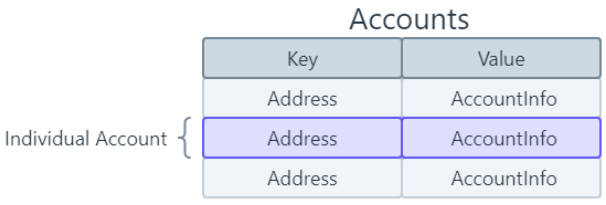
Por otro lado, en Solana, el líder que secuencia las transacciones se elige mediante PoS, un mecanismo que selecciona nodos validadores con base en la cantidad de tokens que poseen y están dispuestos a "apostar" en la red. Este nodo líder organiza las transacciones y reduce la carga de comunicación entre los nodos. Esto mejora la escalabilidad, ya que coordinar menos nodos directamente permite que la red maneje más transacciones por segundo.

De esta manera, debido a que PoS elige un líder para secuenciar las transacciones y PoH permite que la red mantenga el orden de las transacciones de manera independiente, Solana puede manejar una cantidad significativamente mayor de TPS que muchas otras blockchains, incluso sin depender de una fuente centralizada de tiempo.

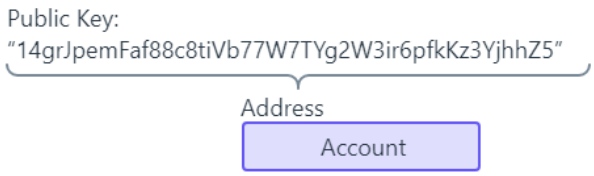
Estructura de la red.

Cuentas.

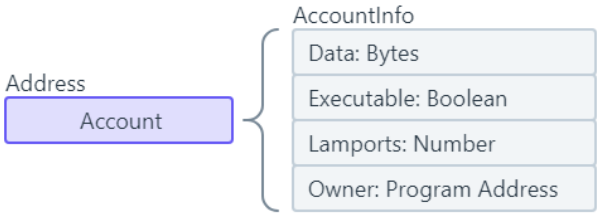
En Solana, todos los datos se almacenan en lo que se denomina "cuentas". La forma en que se organizan los datos en Solana se asemeja a un almacén de clave-valor, donde cada entrada en la base de datos se denomina "cuenta" [6].



Cada cuenta se puede identificar por su dirección única, representada como 32 bytes en el formato Ed25519. PublicKey Puede pensar en la dirección como el identificador único de la cuenta.



Además, las cuentas tienen un tamaño máximo de 10 MB (10 megabytes) y los datos almacenados en cada cuenta en Solana tienen la siguiente estructura conocida como AccountInfo.

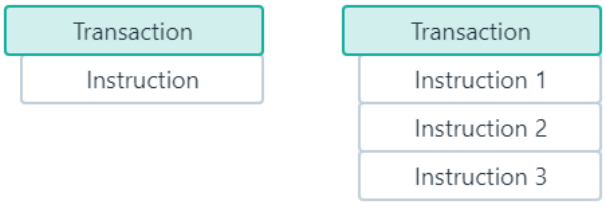


Los campos que incluye son los siguientes:

- data: Una matriz de bytes que almacena el estado de una cuenta. Si la cuenta es un programa (contrato inteligente), almacena el código del programa ejecutable.
- executable: Un indicador booleano que indica si la cuenta es un programa.
- lamports: Una representación numérica del saldo de la cuenta en lamports , la unidad más pequeña de SOL (1 SOL = 1 mil millones de lamports).
- owner: Especifica la clave pública (ID del programa) del programa que posee la cuenta.

Transacciones.

En Solana, se envían transacciones para interactuar con la red. Las transacciones incluyen una o más instrucciones, cada una de las cuales representa una operación específica que se procesará. La lógica de ejecución de las instrucciones se almacena en programas implementados en la red Solana, donde cada programa almacena su propio conjunto de instrucciones [7].



Una transacción es atómica, lo que significa que se completa por completo y se procesan todas las instrucciones correctamente o falla por completo. Si falla alguna instrucción

dentro de la transacción, no se ejecuta ninguna de las instrucciones.

Cada instrucción dentro de una transacción tiene tres componentes clave:

- Programa ejecutor: El programa (smart contract) que debe ejecutarse en respuesta a la instrucción. Esto puede ser un programa estándar o uno personalizado en Solana.
- Cuentas requeridas: La lista de cuentas involucradas en la ejecución de la instrucción. Estas cuentas pueden almacenar datos que el programa necesita leer o modificar.
- Datos: Información adicional necesaria para la ejecución, como parámetros específicos que el programa debe recibir para procesar la instrucción.

Asimismo, se debe tener en cuenta que el tamaño máximo de una transacción en Solana es de 1232 bytes. Esto limita la cantidad de instrucciones que se pueden incluir dentro de una única transacción, así como el tamaño de los datos que se pueden enviar con cada instrucción. Este límite asegura que las transacciones sean eficientes en términos de almacenamiento y procesamiento en la red, contribuyendo a la escalabilidad de Solana.

Particularidades

Sistema de inflación.

El sistema de inflación de Solana está diseñado para apoyar su ecosistema a largo plazo, mientras se ajusta dinámicamente a medida que aumenta el uso de la red y la cantidad de tokens SOL en circulación. Este sistema se estructura a partir de tres parámetros clave: la tasa de inflación inicial, la tasa de desinflación y la tasa de inflación a largo plazo [9].

Al principio, Solana tiene una tasa de inflación relativamente alta, del 8%. Esta tasa significa que, anualmente, la oferta total de SOL en circulación aumenta en un 8%.

Esta emisión inflacionaria está diseñada para recompensar a los participantes que apuestan (stake) sus SOL en la red, ayudando a asegurar la blockchain y validar transacciones. La mayoría de los SOL emitidos a través de la inflación se distribuyen entre los validadores y los delegadores en función de la cantidad de tokens que hayan apostado. Además, la inflación también genera ingresos para los validadores, quienes toman comisiones de los rendimientos generados por el staking de los delegadores.

Por otro lado, La tasa de desinflación indica la velocidad a la que la tasa de inflación disminuye con el tiempo. En el caso de Solana, la inflación disminuye un 15% anualmente a partir del 8% inicial. Este enfoque desinflacionario está diseñado para reducir gradualmente la emisión de nuevos tokens, a medida que la red madura y el uso de Solana se incrementa, lo que disminuye la necesidad de una alta inflación para incentivar la seguridad.

A medida que la red madura y la tasa de inflación disminuye, el sistema está diseñado para alcanzar una tasa de inflación a largo plazo del 1.5%. Esta tasa es lo suficientemente baja como para evitar una devaluación significativa del SOL, pero lo suficientemente alta como para seguir incentivando a los validadores y mantener la seguridad de la red.

A medida que aumenta el uso de la red de Solana, se espera que las tarifas por transacción y otras fuentes de ingresos (como alquileres o quema de tarifas) se conviertan en una parte importante de la economía de la red, lo que podría compensar la disminución de las recompensas inflacionarias.

Unidad 3.
Unidad 4.

Implementación de programas (Contratos inteligentes)

El cargador BPF (Berkeley Packet Filter) es el programa designado como "propietario" de todos los demás programas de la red, excepto los programas nativos. Es responsable de implementar, actualizar y ejecutar programas personalizados.

En solana, se implementa una versión mejorada del BPF, que permite a los desarrolladores desplegar programas.

En cuanto a lenguajes de programación, solana admite la escritura de programas en cadena utilizando el lenguaje de programación Rust [5].

El diseño del proyecto de programas de Solana en Rust sigue una estructura bastante común en proyectos de Rust, pero con algunas particularidades debido al entorno en el que se ejecutan (Solana) y las restricciones de Solana en cuanto a la ejecución de programas.

Estructura del proyecto.

Los proyectos en Rust para Solana siguen una estructura típica de proyecto Rust. Algunos elementos clave incluyen:

- /inc/: Carpeta que probablemente contiene archivos de inclusión o encabezados para el proyecto.
- /src/: Carpeta principal donde se ubica el código fuente de los programas en Rust.
- Cargo.toml: Archivo de configuración del proyecto, que especifica las dependencias y otras configuraciones necesarias para el proyecto en Rust.

Invocación entre programas.

Cuando programas de Solana necesitan invocarse entre sí para acceder a funciones o "asistentes de instrucciones", es posible hacerlo, pero con un detalle importante: no se deben incluir los símbolos de entrada del programa dependiente para evitar conflictos con los símbolos de entrada de tu propio programa.

Dependencias de los programas en Solana Rust.

Los programas Solana Rust tienen una dependencia mínima, que es el paquete solana-program. Sin embargo, hay restricciones debido al entorno donde se ejecutan estos programas, que es el entorno SBF (Solana BPF). Algunas de estas restricciones incluyen:

- Compatibilidad de la arquitectura: Algunos paquetes (o crates) que quieras usar como dependencias pueden requerir una arquitectura que no sea compatible con SBF. Si esto ocurre, no hay una solución sencilla, salvo hacer una bifurcación de ese paquete y modificar las comprobaciones de arquitectura para que sean compatibles con SBF.

- Dependencia en rand: El crate rand, que es común en Rust para generar números aleatorios, no es compatible con el entorno de programas de Solana porque es determinista. En un entorno de blockchain, como Solana, se necesita que los programas sean deterministas para garantizar que siempre produzcan el mismo resultado. Si un crate que necesitas depende de rand, es necesario consultar cómo manejar esta dependencia para que no rompa el determinismo.

- Desbordamiento de pila: Los programas Solana tienen restricciones en el tamaño de la pila, lo que significa que algunas operaciones o crates podrían causar desbordamiento de pila incluso si el código que genera el desbordamiento no es directamente incluido en el programa. Es importante tener esto en cuenta al manejar dependencias.

Discusión.

Referencias.

[1] Solana, "Digital Assets," Solana.com. [Online]. Available: https://solana.com/es/solutions/digital-assets.

[2] A. Yakovenko, "Solana: A new architecture for a high performance blockchain," Solana.com.

[3] Solana, "Blockchain News," Solana.com. [Online]. Available: https://solana.com/news/tag/blockchain.

[4] Cointelegraph, “¿Qué es Solana y cómo funciona?”, Cointelegraph, 2024. [Online]. Available: https://es.cointelegraph.com/news/what-is-solana-and-how-does-it-work.

[5] Solana, “Writing Programs: Rust”, Solana Documentation, 2024. [Online]. Available: https://solana.com/docs/programs/lang-rust.

[6] Solana, “Accounts”, Solana Documentation, 2024. [Online]. Available: https://solana.com/docs/core/accounts.

[7] Solana, “Transactions”, Solana Documentation, 2024. [Online]. Available: https://solana.com/docs/core/transactions.

[8] J. Nwosu, “Solana Explained in 10 Minutes”, YouTube, 2021. [Online]. Available: https://www.youtube.com/watch?v=CJOEkJNz6dU.

[9] Solana, “Inflation Schedule”, *Solana Documentation*, 2024. [Online]. Available: https://solana.com/docs/economics/inflation/inflation-schedule.

Introducción general al protocolo (whitepaper, yellowpaper)
Bondades del protocolo (Bondades, por qué es mejor)
Capitulo preliminar: ¿el protocolo utiliza otras Blockchains?
unidad 2:
particularidades del método del concenso, tokens, red
Unidad 3:
Qué está resolviendo (cálculos, justificación)
Unidad 4:
Resultados propios (despliegue del nodo, profundizara aspectos particulares) enfoque