Universidad de La Habana
Facultad de Matemática y Computación

Parámetros óptimos para redes blockchain de Hyperledger Fabric basada en el análisis de la configuración de transacciones (configtx).

Autor:
Bryan Machín García

Tutores:

Camilo Denis González
Carlos Miguel Legón

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Fecha

github.com/username/repo

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

Índice de figuras

Ejemplos de código

Introducción

Hyperledger Fabric es una plataforma blockchain respaldada por una arquitectura modular que ofrece alta grados de confidencialidad, resiliencia, flexibilidad y escalabilidad. Está diseñada para admitir implementaciones de diferentes componentes y adaptarse a las complejidades que existen en el ecosistema empresarial.

La primera versión de notoria importancia de Hyperledger Fabric luego de su lanzamiento con la versión 1.0 fue Fabric v2.0. Ofrece nuevas características y cambios representativos para usuarios y operadores. Incluye la compatibilidad con nuevos patrones de aplicaciones y privacidad, gobernanza mejorada en torno a contratos inteligentes y nuevas opciones para nodos operativos.

Fabric v2.0 presenta un gobierno descentralizado para contratos inteligentes, con un nuevo proceso para instalar un chaincode en sus peers e instanciarlos en un canal. El nuevo ciclo de vida del chaincode de Fabric permite que varias organizaciones lleguen a un acuerdo sobre los parámetros del chaincode, como la política de respaldo del contrato, antes de que pueda usarse para interactuar con el ledger. El nuevo modelo ofrece varias mejoras con respecto al ciclo de vida anterior:

- **Múltiples organizaciones deben aceptar los parámetros de un chaincode**. En las versiones de lanzamiento 1.x, una organización tenía la capacidad de establecer parámetros de un chaincode, como la política de respaldo, para todos los miembros del canal, que solo tenían el poder de negarse a instalar el chaincode, por lo tanto, no participar en transacciones que lo invoquen. El nuevo ciclo de vida del chaincode de Fabric es más flexible, admite tanto el modelo de ciclo de vida anterior, como modelos descentralizados que requieren suficiente número de organizaciones para acordar una política de respaldo y otros detalles antes de que el chaincode se convierta en activo en un canal.
- **Proceso de actualización de chaincode más deliberado**. Anteriormente la transacción de actualización podía ser emitida por una sola organización, creando un riesgo para un miembro del canal que aún no había instalado el nuevo

chaincode. El nuevo modelo permite que el chaincode se actualice solo después de que un número suficiente de organizaciones han aprobado la actualización.

- **Actualizaciones más sencillas de políticas de respaldo y recopilación de datos privados**. Se permite cambiar una política de respaldo o una configuración de recopilación de datos privados sin tener que volver a empaquetar o instalar el chaincode. Los usuarios también pueden aprovechar una nueva política de respaldo predeterminada que requiere el respaldo de una mayoría de organizaciones en el canal. Esta política se actualiza automáticamente cuando se agregan o eliminan organizaciones del canal.

- **Paquetes de chaincodes inspeccionables**. El chaincode se empaqueta en archivos .tar fáciles de leer, que hace más factible su inspección y coordinación de la instalación en múltiples organizaciones.

- **Iniciar múltiples chaincodes en un canal usando un paquete**. El ciclo de vida anterior definía cada chaincode en el canal usando un nombre y una versión que se especificó en su instalación. Ahora existe la posibilidad de usar un solo paquete de chaincodes y desplegarlo varias veces con diferentes nombres en el mismo canal o en diferentes canales.

Los mismos métodos descentralizados para llegar a un acuerdo que sustentan la nueva gestión del ciclo de vida del chaincode pueden usarse también en su propia aplicación para garantizar que las organizaciones den su consentimiento a las transacciones de datos antes de que se realicen escrituras en el ledger.

- **Comprobaciones automatizadas**. Las organizaciones pueden agregar comprobaciones automáticas a las funciones del chaincode para validar información adicional antes de aprobar una propuesta de transacción.

- **Acuerdo descentralizado**. Las decisiones humanas se pueden modelar en un proceso de chaincode que abarca múltiples transacciones. El chaincode puede requerir que los actores de varias organizaciones indiquen sus términos y condiciones de acuerdo en una transacción en el ledger. Luego, una propuesta final de chaincode puede verificar que las condiciones de todas las transacciones individuales se cumplen y establecer la transacción comercial con carácter definitivo en todos los miembros del canal.

Fabric v2.0 también permite nuevos patrones para trabajar y compartir datos privados, sin el requisito de crear recopilaciones de datos privados para todas las combinaciones de miembros del canal que deseen realizar transacciones. Específicamente,

en lugar de compartir datos privados dentro de una colección de varios miembros. Es posible compartir datos privados entre colecciones, donde cada colección puede incluir una sola organización, o tal vez una sola organización junto con un regulador o auditor.

- **Compartir y verificar datos privados**. Cuando los datos privados se comparten con un miembro del canal que no es miembro de una colección, o compartida con otra colección de datos privados que contiene uno o más miembros del canal (escribiendo una clave para esa colección), las partes receptoras pueden utilizar la API del chaincode $GetPrivateDataHash()$ para verificar que los datos privados coincidan con los $hash$ en la cadena que se crearon a partir de datos privados en transacciones anteriores.
- **Políticas de aprobación a nivel de colección**. Las colecciones de datos privados ahora se pueden definir opcionalmente con una política de aprobación que anula la política de aprobación a nivel de chaincode para las claves dentro de la colección. Se puede usar para restringir qué organizaciones pueden escribir datos en una colección. Se puede concebir un chaincode que requiere el respaldo de la mayoría de las organizaciones, pero para cualquier transacción determinada, puede necesitar dos organizaciones transaccionales para respaldar individualmente su acuerdo en sus propias colecciones de datos privados.
- **Recopilaciones implícitas por organización**. Si se desea utilizar patrones de datos privados por organización, no se necesita definir las colecciones al implementar chaincode. Las colecciones se pueden usar sin ninguna definición inicial.

La función de lanzador de chaincode externo permite a los operadores crear y lanzar chaincode con la tecnología de su elección. No se requiere el uso de constructores y lanzadores externos ya que el comportamiento predeterminado construye y ejecuta el chaincode de la misma manera que las versiones anteriores con la API de Docker.

- **Elimina la dependencia del demonio de Docker**. Las versiones anteriores de Fabric requerían que los pares tuvieran acceso a un $Dockerdaemon$ para construir y lanzar chaincode, algo que puede no ser deseable en entornos de producción debido a los privilegios requeridos por el proceso de pares.
- **Alternativas a los contenedores**. Ya no es necesario ejecutar chaincode en contenedores Docker, y puede ejecutarse en el entorno elegido por el operador (incluidos los contenedores).

- **Ejecutables del constructor externo**. Un operador puede proporcionar un conjunto de ejecutables del constructor externo para anular cómo un peer construye y lanza chaincode.
- **Chaincode como un servicio externo**. Tradicionalmente, los chaincodes son lanzados por el peer y luego se conectan de nuevo al peer. Ahora es posible ejecutar chaincode como un servicio externo, por ejemplo, en un pod de Kubernetes, que un peer puede conectarse y utilizar para la ejecución de chaincode.

Cuando se utiliza una base de datos de estado externa $CouchDB$, los retrasos de lectura durante las fases de aprobación y validación han representado un cuello de botella en el rendimiento. Con Fabric v2.0, una nueva caché reemplaza muchas de estas costosas búsquedas con lecturas rápidas de caché local. El tamaño de caché se configura mediante la propiedad $cacheSize$ en $core.yaml$.

Hyperledger Fabric posee un diseño atractivo para uso empresarial, producto a su adaptabilidad en escenarios de consensos y políticas de gobierno para varias entidades. Resulta esencial el rendimiento de la plataforma en la transacción de información, lo cual nos motiva a realizar un estudio que posibilite obtener un conjunto de configuraciones óptimas.

Capítulo 1

Estado del Arte de la plataforma blockchain Hyperledger Fabric

Hyperledger Fabric es una de las plataformas blockchain más populares, administrada por Linux Foundation. Se basa en una plataforma privada donde solo los usuarios autenticados participan en ella. Difiere de las plataformas blockchain públicas que posibilitan la unión de cualquier usuario a la red. Además, Fabric presenta una arquitectura de ejecución, orden y validación que supera los límites de la arquitectura de orden y ejecución anterior []. Esto mejora sustancialmente la escalabilidad de rendimiento en redes blockchain con un número elevado de Peers, lo que permite a Fabric ser competente en Global Trade Digitalization [], SecureKey [] y Everledger [].Constituye la primera plataforma blockchain que admite contratos inteligentes creados en lenguajes de programación de uso general como Java, Golang y Node.js; siendo factible para la mayoría de las empresas en el desarrollo de los contratos inteligentes, sin necesidad de capacidad adicional para aprender lenguajes específicos de dominio restringidos, conocidos por sus siglas DSL.

Fabric es compatible con protocolos de consenso conectables que permiten a la plataforma personalizarse de manera eficaz de acuerdo al caso de uso y modelos de confianza de los entes que la integran.

Capítulo 2

Propuesta

Capítulo 3

Detalles de Implementación y Experimentos

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones