ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

IMPLEMENTACIÓN DE UNA ARQUITECTURA BASADA EN BLOCKCHAIN PARA SECUENCIAR LA OCURRENCIA DE EVENTOS EN BASES DE DATOS DISTRIBUIDAS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SOFTWARE MENCIÓN SEGURIDAD

GUSTAVO ALFONSO ALCÍVAR RODRÍGUEZ
gustavo.alcivar@epn.edu.ec

Director: Dr. Denys Alberto Flores Armas
denys.flores@epn.edu.ec

2021

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación IMPLEMENTACIÓN DE UNA ARQUITECTURA BASADA EN BLOCKCHAIN PARA SECUENCIAR LA OCURRENCIA DE EVENTOS EN BASES DE DATOS DISTRIBUIDAS desarrollado por ALCÍVAR RODRÍGUEZ GUSTAVO ALFONSO, estudiante de la MAESTRÍA EN SOFTWARE, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

Digitally signed by DENYS ALBERTO FLORES ARMAS
DN: cn=DENYS ALBERTO FLORES ARMAS, serialNumber=231120155658, ou=ENTIDAD DE CERTIFICACION DE INFORMACION, o=SECURITY DATA S.A. 2, c=EC
Date: 2021.09.15 10:43:46 -05'00'

Dr. Denys Alberto Flores Armas

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, GUSTAVO ALFONSO ALCÍVAR RODRÍGUEZ, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

GUSTAVO ALFONSO ALCÍVAR RODRÍGUEZ

DEDICATORIA

Dedico esta tesis principalmente a mis padres, quienes me motivan a siempre buscar nuevas metas y me brindan su apoyo incondicional para conseguirlas.

AGRADECIMIENTO

Agradezco a mi madre Carmen Maricela, a mi padre Gustavo Francisco, a mi hermana María Alejandra y a mi hermano Luis Sebastián por brindarme su apoyo durante todo este tiempo y animarme diariamente para lograr cumplir esta meta.

Agradezco a mi tutor Denys Flores por haberme guiado, motivado y brindado su ayuda durante este proceso.

Agradezco a la Escuela Politécnica Nacional por haberme dado la oportunidad de crecer personal, académica y profesionalmente mediante esta maestría.

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS i
LISTA DE TABLAS ii
LISTA DE ANEXOS iii
RESUMEN iv
ABSTRACT v

1. INTRODUCCIÓN 1
1.1. ESPECIFICACIÓN DEL PROBLEMA 1
1.2. PREGUNTA DE INVESTIGACIÓN 3
1.3. OBJETIVO GENERAL 3
1.4. OBJETIVOS ESPECÍFICOS 3
1.5. ALCANCE 3
1.6. METODOLOGÍA 4
2. MARCO TEÓRICO 7
2.1. AUDITORÍA DE BASES DE DATOS 7
2.1.1. Auditoría proactiva 8
2.1.2. Requerimientos de auditoría de bases de datos 8
2.1.3. Concurrencia de transacciones 9
2.2. BLOCKCHAIN 10
2.2.1. Distributed Ledger Technology (DLT) 10
2.2.2. Forma de trabajo de blockchain 10
2.2.3. Protocolo de consenso 11
2.2.4. Practical Byzantine Fault Tolerance (PBFT) 12
2.2.5. Hyperledger Sawtooth 14
2.3. TRABAJOS RELACIONADOS 14
3. IMPLEMENTACIÓN 15
3.1. DISEÑO DE LA SOLUCIÓN 16
3.1.1. Elección de la tecnología para la base de datos 16
3.1.2. Elección de la tecnología para la blockchain 16
3.1.3. Esquema de la solución 16
3.1.4. Fase de generación 20

3.1.5. Fase de recolección 21
3.1.6. Fase de preservación 23
3.2. EVALUACIÓN DE LA SOLUCIÓN 25
3.2.1. Evaluación de rendimiento 25
3.2.2. Consistencia de tiempo 34
4. RESULTADOS Y DISCUSIÓN 36
4.1. RESULTADOS 36
4.2. DISCUSIONES 36
5. CONCLUSIONES Y RECOMENDACIONES 37
5.1. CONCLUSIONES 37
5.2. RECOMENDACIONES 38
REFERENCIAS BIBLIOGRÁFICAS 39
ANEXOS 42

LISTA DE FIGURAS

FIGURA 1 - ALGORITMO DE PBFT 13
FIGURA 2 - ESQUEMA DE LA SOLUCIÓN 17
FIGURA 3 - SCRIPT EJEMPLO PARA HABILITAR SQL SERVER SERVICE BROKER 19
FIGURA 4 – ALGORITMO DE GENERACIÓN 20
FIGURA 5 – ALGORITMO DE RECOLECCIÓN 22
FIGURA 6 - SCRIPT EJEMPLO DEL TRIGGER DE RECOLECCIÓN DE DATOS DE AUDITORÍA 22
FIGURA 7 - ALGORITMO DE PERSERVACIÓN 24
FIGURA 8 – COMUNICACIÓN ENTRE SERVICE BROKER Y EL CLIENTE BLOCKCHAIN 24
FIGURA 9 - COMUNICACIÓN ENTRE EL CLIENTE BLOCKCHAIN Y LA BLOCKCHAIN 24
FIGURA 11 – USO DE PROCESADOR CON 100 TRANSACCIONES 26
FIGURA 12 - BYTES DE RED TRANSMITIDOS CON 100 TRANSACCIONES 27
FIGURA 13 - OPERACIONES DE LECTURA / ESCRITURA EN DISCO CON 100 TRANSACCIONES 28
FIGURA 14 - USO DE PROCESADOR CON 500 TRANSACCIONES 29
FIGURA 15 - BYTES DE RED TRANSMITIDOS CON 500 TRANSACCIONES 30
FIGURA 16 - OPERACIONES DE LECTURA / ESCRITURA EN DISCO CON 500 TRANSACCIONES 31
FIGURA 17 - USO DE PROCESADOR CON 1000 TRANSACCIONES 32
FIGURA 18 - BYTES DE RED TRANSMITIDOS CON 1000 TRANSACCIONES 33
FIGURA 19 - OPERACIONES DE LECTURA / ESCRITURA EN DISCO CON 1000 TRANSACCIONES 34

LISTA DE TABLAS

TABLA 1 - LÍNEAS GUÍA DE LA METODOLOGÍA DESIGN SCIENCE RESEARCH 5
TABLA 2 – TÉCNICAS PARA TRATAR LA CONCURRENCIA 9
TABLA 3 - PROTOCOLOS DE CONSENSO 11
TABLA 4 - CARACTERÍSTICAS DE HYPERLEDGER SAWTOOTH 14
TABLA 5 - CARACTERÍSTICAS DE LAS MÁQUINAS VIRTUALES DE GCP 18
TABLA 6 - MUESTRA DE LOS TIEMPOS RECOLECTADOS 35

LISTA DE ANEXOS

ANEXO I – GUÍA TÉCNICA DE IMPLEMENTACIÓN DE LA SOLUCIÓN.....43
ANEXO II – PASOS PARA CONFIGURAR EL TABLERO DE GRAFANA.....60
ANEXO III – CAPTURA DE PANTALLA DEL TABLERO DE GRAFANA. 65
ANEXO IV – CAPTURA DE PANTALLA DE LA APLICACIÓN FRONTEND DESARROLLADA.....66

RESUMEN

La auditoría de bases de datos distribuidas podría ser un gran desafío debido a la dificultad de determinar el momento de ocurrencia de una transacción [1]. De hecho, cuanto mayor sea el número de transacciones distribuidas en un período corto de tiempo, más difícil será ordenar su ocurrencia. A pesar de la importancia de resolver este problema, se han realizado muy pocas investigaciones en los últimos años, estas se han centrado principalmente en mejorar el rendimiento de las bases de datos más que en su seguridad. Mientras tanto, la tecnología blockchain emergente en los últimos años puede sugerir soluciones elegantes para garantizar la integridad transaccional entre los nodos de bases de datos distribuidas. Sin embargo, esta tecnología no es muy conocida dentro de las organizaciones y, en ocasiones, su implementación se considera difícil debido a la falta de pautas de implementación. En este trabajo, se propone una arquitectura basada en blockchain centrada en la secuenciación transaccional de eventos en un entorno distribuido. Se usa una tecnología blockchain de código abierto para demostrar su funcionalidad con fines de auditoría. Mejoramos los controladores de auditoría propuestos en investigaciones anteriores para que sean compatibles con blockchain. Finalmente, se evalúa el rendimiento de la arquitectura y su consistencia distribuida utilizando una alta carga de trabajo transaccional dentro de una plataforma comercial en la nube. Además de proporcionar pautas de implementación detalladas utilizando la tecnología blockchain de código abierto, la principal contribución es demostrar su aplicación para generar, recolectar y preservar pistas de auditoría en una base de datos distribuida. Los resultados también demuestran su excelente desempeño durante una alta carga de trabajo transaccional, ya que el 90 por ciento de las transacciones se registran en menos de 1 minuto.

Palabras clave: Blockchain, Base de datos Distribuida, Hyperledger Sawtooth, Auditoría, Concurrencia

ABSTRACT

Auditing distributed databases could be very challenging due to the difficulty of determining the time of occurrence of any transaction [1]. In fact, the higher the number of distributed transactions in a short period of time, the harder it is to order their occurrence. Despite the importance of solving this problem, very little research has been done in the last years, which has been mostly focused on improving databases' performance rather than their security. Meanwhile, the emerging blockchain technology in recent years has devised elegant solutions to guarantee transactional integrity between distributed database nodes. However, this technology is not very well known within organizations, and sometimes its implementation is seen as difficult due to the lack of deployment guidelines. In this paper, we propose a blockchain-based architecture focused on transactional sequencing of events in a distributed environment. We use an open source blockchain technology to demonstrate its functionality for auditing purposes. We enhance audit controllers proposed in previous research to make them blockchain-compliant. Finally, we evaluate the architectural performance and its distributed consistency using high transactional workload within a commercial Cloud platform. Besides providing detailed implementation guidelines using open source blockchain technology, our main contribution is demonstrating its application for generating, collecting and preserving audit trails in a distributed database. Our results also demonstrate its excellent performance during high transactional workload as 90 per cent of transactions commit in less than one minute.

Index Terms: Blockchain, Distributed Databases, HyperledgerSawtooth, Audit, Concurrency.

1. INTRODUCCIÓN

En esta sección se introduce el problema que se quiere resolver, se delimita el alcance de la propuesta, se definen los objetivos planteados y se establece la metodología que se utilizó en la investigación.

1.1. Especificación del problema

En la actualidad, un gran número de empresas cuenta con sistemas informáticos que se conectan a bases de datos, las cuales se encuentran en servidores. En ocasiones existe tal número de conexiones simultáneas, que es necesario implementar bases de datos distribuidas para evitar la sobrecarga ya que, en cada conexión es posible que se ejecuten una o más transacciones u operaciones de consulta o modificación sobre los datos.

En las bases de datos distribuidas, un problema que se presenta es, por ejemplo, si se realizan dos o más transacciones simultáneamente en diferentes servidores, podría resultar complejo determinar cuál de las transacciones se realizó primero.

Determinar el orden de las transacciones en una base de datos distribuida permite generar una línea de tiempo global, es decir, un registro que incluya a todos los servidores y que muestre la secuencia de las operaciones realizadas sobre los datos, este ordenamiento global resulta de suma importancia al momento de realizar auditorías, ya que el orden de las transacciones podría considerarse como una pista o evidencia de un delito, por ello, las bases de datos cuentan con tablas de auditoría, en las que se registran las transacciones y la marca de tiempo en la que cada una de ellas se realizó, sin embargo, en ambientes distribuidos, es posible que existan varios registros con una misma marca de tiempo y resulte complejo determinar la secuencia correcta, como se muestra en [1]. El mencionado problema se puede mitigar configurando los servidores de manera que guarden los registros con más precisión de tiempo, sin embargo, dos problemas adicionales que podrían ocurrir son: Primero, si el reloj local de los servidores no está sincronizado con algún servicio central, puede darse el caso de que cada uno tenga configurado una fecha u hora diferentes, lo que provocaría registros con valores de tiempo no confiables y segundo, que alguno de los servidores se vea comprometido y los registros de auditoría sean modificados de forma no autorizada, alterando la secuencialidad de las transacciones.

Los problemas de secuencialidad de transacciones y repudiación son un tema recurrente en bases de datos distribuidas [2], y en el presente proyecto se plantea una nueva forma de hacer frente a dichos problemas mediante una arquitectura que aproveche los beneficios de blockchain y que además ayude al sistema de control interno de las compañías que cuenten con bases de datos distribuidas.

Blockchain, gracias a su estructura y modo distribuido de funcionar, ayudaría a generar un registro ordenado cronológicamente de transacciones como se puede ver en [3], de este modo, el sello de tiempo con el que se registre una transacción no sería concluyente para determinar la secuencialidad, y más bien, el ordenamiento estaría dado por la cadena de bloques, es decir, blockchain puede aportar en la generación y recolección de datos para auditoría de sistemas.

En 2015 la agencia de inteligencia de Estados Unidos mencionaba que uno de los grandes problemas de seguridad de estos años, sería la manipulación de los datos [4], tal manipulación de datos podría darse por parte del responsable de una operación ilegítima sobre la base de datos, el cual tenga acceso a realizar modificaciones sobre los registros, de modo que elimine alguna evidencia que lo pueda delatar. En dicho caso, se habla de un problema que afecta a la propiedad de no repudio de los sistemas, tal problema hace referencia a que una persona niegue haber realizado una determinada acción. En este sentido, la utilización de blockchain aporta a la mitigación de problemas de repudiación, ya que, quedaría registrado en la cadena de bloques el origen y responsable de toda transacción, por lo tanto, no sería posible evadir la responsabilidad o, dicho de otra manera, se garantiza la preservación de la evidencia.

De lo expuesto en los últimos párrafos, se concluye que blockchain puede cubrir las tres etapas fundamentales de una auditoría: **Generación**, **recolección** y **preservación** de datos de auditoría. Con esto a su vez, se garantiza el no repudio y se contribuye al control interno.

Existen varias soluciones que implementan blockchain a nivel empresarial y, que proveen mecanismos para interactuar con esta de manera bastante accesible, una de ellas, y la que se utilizó en el presente proyecto, es Hyperledger Sawtooth [5], dicho proyecto se encuentra en un estado maduro y su implementación ya incorpora los mecanismos necesarios para el consenso entre los nodos de la red y la replicación de información.

El resultado de este trabajo puede ser aplicado en cualquier empresa en cuya arquitectura tecnológica tenga bases de datos distribuidas y deseé contar con un mecanismo que ayude al momento de realizar auditorías, mitigue el problema de repudiación y mejore el sistema de control interno de la compañía.

1.2. Pregunta de investigación

¿Es factible utilizar la tecnología de blockchain como sistema para la generación de datos de auditoría en donde se asegure la secuencialidad y el no repudio de transacciones en bases de datos distribuidas?

1.3. Objetivo general

Implementar una arquitectura basada en blockchain que permita secuenciar la ocurrencia de eventos en bases de datos distribuidas.

1.4. Objetivos específicos

- Identificar requerimientos funcionales mínimos para monitorear la ocurrencia de eventos en una base de datos distribuida.
- Implementar una arquitectura, utilizando la tecnología blockchain, para secuenciar la ocurrencia de eventos en una base de datos distribuida, mediante la creación de una línea de tiempo global de tales eventos.
- Evaluar el desempeño de la arquitectura propuesta considerando el uso de recursos.
- Plantear trabajos futuros para la aplicación de arquitecturas basadas en blockchain en ámbitos diferentes a las auditorías en bases de datos distribuidas.

1.5. Alcance

Se consideran como parte del proyecto, la implementación y configuración de la blockchain desde cero, así como también el despliegue de la solución desarrollada en un ambiente en la nube. La implementación se la realiza en máquinas virtuales en la plataforma de Google Cloud y se ejecutan pruebas sobre dicha implementación desde otra máquina virtual en la misma plataforma mediante la aplicación Apache JMeter [6].

Para la base de datos distribuida se utilizó la tecnología de Microsoft Distributed Transaction Coordinator (MSDTC) con 4 servidores conectados una misma red.

Las pruebas y toma de muestras de rendimiento y funcionalidad de la solución, se la realizan en la arquitectura en la nube implementada para este proyecto y no se realizan pruebas sobre bases de datos en servidores físicos de ninguna empresa en particular.

1.6. Metodología

En el presente proyecto se utilizó la metodología "Design Science Research" [7], se decidió trabajar sobre esta metodología debido a que esta permite medir de forma cuantitativa los resultados y busca generar nuevo conocimiento, en el caso particular de este proyecto, se ha generado conocimiento que puede ser utilizado en auditorías de bases de datos.

Design Science Research se puede aplicar a varias ramas de la ciencia y de manera general, parte de un problema, luego, define los objetivos de la solución al problema, a continuación, se diseña y desarrolla la solución, se realiza una demostración de esta, se lleva a cabo una evaluación y, finalmente, se debe realizar una socialización de la solución encontrada. En la presente investigación, estas etapas se detallan a continuación:

- **Problema:** Ordenamiento de transacciones en bases de datos distribuidas.
- **Objetivos de la solución:** Implementar una arquitectura con la utilización de blockchain que permita el ordenamiento de las transacciones en bases de datos distribuidas.
- **Diseño, desarrollo, demostración de la solución:** Para estas etapas, se utilizó el proyecto de blockchain Hyperledger Sawtooth y se detalla en el capítulo de implementación.
- **Evaluación:** Para esta etapa se utilizó el software Apache JMerter y se detalla en el capítulo de evaluación.
- **Socialización:** Esta etapa se cumple con la elaboración de este documento y el artículo generado del mismo.

Para llevar a cabo estas etapas, la metodología "Design Science Research" considera 7 líneas guía [7], las cuales se presentan en la Tabla 1.

Tabla 1 - Líneas guía de la metodología Design Science Research

Línea guía	**Descripción**
Diseño de un artefacto	Esta metodología debe producir un artefacto, en este caso, es el modelo de una arquitectura que integra una base de datos distribuida con una blockchain para secuenciar los eventos en este tipo de base de datos.
Problema relevante	El objetivo de esta metodología es desarrollar una solución basada en la tecnología para solventar un problema empresarial de importancia, en este caso, se apunta hacia el problema de la concurrencia de eventos en bases de datos distribuidas.
Diseño de la evaluación	En Design Science Research es necesario que sea posible evaluar el artefacto diseñado, en la presente investigación, la evaluación se enfoca en el rendimiento de la arquitectura propuesta mediante la utilización de la herramienta Apache JMeter.
Contribuciones de la investigación	En la metodología seleccionada, se debe contribuir al área que se investiga, en el caso de esta investigación, se pretende aportar en el ámbito de las auditorías de bases de datos, el control interno y la repudiación.
Rigor de la investigación	La implementación de la arquitectura se encuentra detallada en el Anexo I y en un repositorio en GitHub generado para el proyecto. Para la evaluación de la solución, se utilizó el software Apache JMeter y los ficheros de configuración de las pruebas se encuentran publicados en el repositorio de

	la solución, lo cual permite que tanto la implementación como las pruebas y evaluación puedan ser replicadas en diferentes escenarios.
Diseño como proceso de búsqueda	La metodología indica que es necesario utilizar los medios disponibles para alcanzar los fines deseados, satisfaciendo siempre los requerimientos de los problemas planteados. En el caso particular de este trabajo, el fin deseado es generar un registro ordenado de las transacciones ejecutadas en una base de datos, para ello se utilizó una plataforma en la nube y un proyecto de blockchain de uso empresarial y de código abierto.
Comunicación de la investigación	En Design Science Research, se debe presentar tanto a personas con conocimientos en tecnología, como a personas orientadas a la gestión. En este caso particular, se lo hace mediante el presente trabajo y un artículo científico.

Para cumplir con las líneas guía planteadas por la metodología, en primer lugar, a través de la búsqueda sistemática de material bibliográfico, se determinaron los requerimientos funcionales para sistemas de auditoría y se indagó en las soluciones de blockchain que se están utilizando a nivel empresarial.

Una vez seleccionada la blockchain a utilizar, se modeló una arquitectura que permite integrar una base de datos con la blockchain seleccionada. Tal arquitectura permite que, cada vez que en la base de datos se ejecuta una transacción, se añade la información de dicha transacción a un bloque de la blockchain, además, previo al registro en la blockchain, se agrega información de auditoría importante como el usuario que ejecutó la transacción, el servidor en el que se ejecutó y los tiempos en los que el cliente realizó la transacción, llegó a la base de datos y se registró en la blockchain.

Luego de la implementación de la solución, se realizaron experimentos sobre la misma considerando diferente volumen de transacciones para obtener datos referentes al uso de recursos de los servidores con la solución activa e inactiva, para posteriormente analizar los datos y determinar la aplicabilidad de la arquitectura propuesta.

La arquitectura propuesta cumple el objetivo planteado ya que genera una línea de tiempo de las transacciones realizadas en todos los servidores y conserva el orden en el que las transacciones fueron ejecutadas.

2. MARCO TEÓRICO

En esta sección se tratarán, a nivel teórico, todos los conceptos involucrados en la investigación.

El presente trabajo se centra en las auditorías de bases de datos distribuidas aplicando la tecnología de blockchain, por ello, el estudio teórico se divide en estos dos grandes componentes.

2.1. Auditoría de bases de datos

Una auditoría es un proceso de validación del cumplimiento de parámetros establecidos en una actividad determinada. En bases de datos, la auditoría involucra observación de la base de datos para estar al tanto de las acciones que realizan los usuarios sobre los datos. Los administradores y consultores de base de datos suelen establecer auditorías con fines de seguridad, por ejemplo, para garantizar que quienes no tienen permiso para acceder a la información no accedan a ella [8].

Yang, en [9] dice que, en una base de datos no existe seguridad si no existe auditoría, por ello, seguridad y auditoría deben ser implementados conjuntamente, además, las auditorías deben jugar un papel central para asegurar el cumplimiento de las políticas y normas de una compañía, principalmente porque en una auditoría se examinan las acciones y conductas de los individuos.

Un aspecto importante en una auditoría es la calidad de los datos, y esta dependerá de la forma en la que se realice la escritura de los datos y está altamente relacionada con los mecanismos de control interno de las compañías [10], por ello, hoy en día se busca un

enfoque de auditoría proactiva que implica que los sistemas recolecten evidencia en tiempo real.

2.1.1. Auditoría proactiva

La auditoría proactiva se refiere a que no sea necesario utilizar herramientas externas para generar datos de auditoría, y que más bien, los propios sistemas sean quienes generen estos datos durante su funcionamiento regular [11].

Gracias a este enfoque proactivo, si se identifica un problema durante el funcionamiento del sistema, se puede realizar un análisis de las causas de este en tiempo real y se puedan tomar los correctivos necesarios.

2.1.2. Requerimientos de auditoría de bases de datos

Un término altamente ligado a la auditoría es la informática forense, en esta existen 4 principios básicos que pueden ayudar al momento de realizar auditorías, en particular, el tercer principio. Estos principios están listados en la guía de buenas prácticas para la evidencia digital de la ACPO [12] y el principio 3 dicta lo siguiente:

"Todas las operaciones realizadas sobre la evidencia digital deben quedar registradas, de modo que, una persona o entidad independientes puedan replicar tales operaciones y llegar al mismo resultado".

El principio antes mencionado implica considerar la marca de tiempo de una transacción como un requisito para monitorear el comportamiento de la información. La marca de tiempo permite construir una línea de tiempo de los eventos de manipulación de datos (DML) realizados sobre la base de datos, el cual puede servir no solo para explicar su ordenamiento global, sino también para generar, recolectar y conservar pistas de auditoría [13].

La **generación**, **recolección** y **preservación** de datos de auditoría son precisamente los requerimientos que se deben cubrir cuando se realiza una auditoría de base de datos y podrían considerarse como las fases a cumplir para generar datos de auditoría en una base de datos.

- **Fase de generación:** Esta fase se refiere al mecanismo mediante el cual los registros de la base de datos son generados, este podría ser a través de la interacción de la

base de datos con aplicaciones externas o mediante la interacción desde el propio sistema de gestión de la base de datos.

- **Fase de recolección:** Se refiere a como se recolectan los datos de auditoría, en la solución propuesta en el presente trabajo, esta fase se la realiza mediante TRIGGERS en la base de datos.
- **Fase de preservación:** Se refiere a como se asegura que los datos recolectados en las fases previas se conserven y se asegure que no sean modificados.

En el presente trabajo se cubren estos requerimientos considerando un ambiente distribuido y, el principal problema en tal tipo de ambientes es la concurrencia de las transacciones, ya que, al ser un ambiente distribuido en varios nodos o equipos, las transacciones se ejecutan desde varias fuentes al mismo tiempo.

2.1.3. Concurrencia de transacciones

Cuando se habla de ambientes distribuidos, es altamente probable que se presente el problema de la concurrencia. Existen técnicas específicas para tratar el problema de la concurrencia en bases de datos distribuidas, algunas se muestran en la Tabla 2.

Tabla 2 – Técnicas para tratar la concurrencia

Técnica	**Descripción**
Two-Phase Locking – 2PL	Podría considerarse como el primer control de concurrencia con serialización, se desarrolla en dos fases: en la primera, la transacción aplica un bloqueo sobre el registro que necesita acceder para realizar la afectación; y en la segunda fase, se libera el bloqueo una vez que se finaliza la afectación sobre el registro.
Timestamp Ordering	En este protocolo, simplemente se realiza un ordenamiento de las transacciones tomando en cuenta la marca de tiempo en la que se registraron.
Optimistic concurrency control – OOC	Considera que todas las transacciones son ejecutadas de manera serial y, toma el orden en que las afectaciones a la base de datos se realizaron como el orden legítimo.
Deterministic	Todos los nodos reportan las transacciones a un único servidor central, el cual es el responsable de determinar el ordenamiento, de este modo, se evita

	que los servidores deban coordinarse entre sí.

Extraído de [2] [14]

Las técnicas de la Tabla 2 ayudan a resolver el problema del ordenamiento de las transacciones, sin embargo, son susceptibles a que se modifiquen los datos de forma no autorizada sin dejar algún rastro que pueda servir para determinar un responsable, aspecto que puede ser solventado con la aplicación de la tecnología blockchain.

2.2. Blockchain

Antes de mencionar lo que es blockchain, en necesario definir lo que es Distributed Ledger Technology.

2.2.1. Distributed Ledger Technology (DLT)

Distributed Ledger Technology es una base de datos descentralizada que se encuentra distribuida entre varios participantes [15]. En la cual, no existe una entidad central que se encargue de la verificación de los registros de la base de datos, la misma se realiza mediante un consenso entre los participantes, lo cual aumenta la transparencia y la validez de los registros además de que dificulta la manipulación no autorizada de los datos. Las diferentes formas en la que los participantes pueden llegan a un acuerdo sobre la validez de un registro se denomina "protocolo de consenso".

Blockchain es un tipo particular de DLT en donde la única operación permitida es la adición de nuevos registros. Es la tecnología que está detrás de muchas cryptomonedas y es una DLT que ha sido ampliamente estudiada [16]. El hecho de que no se puedan actualizar o eliminar registros permite que se pueda tener constancia de cada una de las operaciones realizadas sobre la base de datos.

2.2.2. Forma de trabajo de blockchain

En blockchain, la información se agrupa en bloques, cada uno de ellos se enlaza con el anterior mediante la función hash [17], esto permite asegurar la integridad de la información ya que, si se realiza una modificación en los datos de uno de los bloques, su valor hash cambiará, de modo que ya no coincidirá con el valor registrado en el bloque posterior, rompiendo la cadena de bloques y perdiendo su validez. Este hecho por sí solo no asegura que los datos no sean modificados, ya que, una vez realizada una alteración de la información, es posible volver a calcular y actualizar el valor hash del bloque

alterado y recursivamente los valores hash de todos los bloques posteriores, de modo que la blockchain mantenga su consistencia. Es por ello por lo que, el verdadero potencial de blockchain se presenta cuando se tiene un escenario distribuido y se considera el concepto de “algoritmo de consenso” descrito previamente en este capítulo para determinar si un bloque es válido y debe ser agregado a la cadena.

Se ha mencionado que en DLT, la aceptación o rechazo de un registro se hace mediante un protocolo de consenso, concepto que se define a continuación.

2.2.3. Protocolo de consenso

Es un mecanismo, representado mediante un algoritmo, que permite que varios dispositivos o usuarios se coordinen en una configuración distribuida [18].

En el contexto de la tecnología de registro distribuido, el protocolo de consenso es el mecanismo con el cual se determina si un bloque (conjunto de transacciones) es válido y puede ser agregado a la base de datos distribuida.

Algunos protocolos de consenso se muestran en la Tabla 3.

Tabla 3 - Protocolos de consenso

Protocolo	**Descripción**
Practical Byzantine Fault Tolerance (PBFT)	En este algoritmo se consideras un nodo líder, quien será el encargado de distribuir la transacción al resto de nodos y para validar la transacción, se sigue un determinado algoritmo que será explicado más adelante en este capítulo.
Proof of Work (PoW)	El primer participante en completar una determinada prueba computacional, como resolver un problema criptográfico es quien agrega el nuevo bloque de transacciones. Este es el método aplicado por Bitcoin [19].
Proof of Stake (PoS)	Cada participante adquiere un determinado peso que se puede obtener de acuerdo con una cantidad de criptomonedas o mediante el poder computacional del participante.

	Cada participante vota sobre la validez del bloque tomando en cuenta el peso de cada uno y se llega a un consenso para agregar el bloque.
Delegated Proof of Stake (DPoS)	Similar a PoS, se diferencia en que, en este caso, en lugar de votar sobre la validez del bloque, se realiza una votación para elegir al nodo que será en encargado de verificar que el bloque sea correcto.
Proof of Elapsed Time (PoET)	Introducido por Hyperledger Sawtooth, una fuente confiable genera un tiempo de espera aleatorio para cada participante y el participante con el menor tiempo es quien decide sobre la validez del bloque. Este método optimiza el uso de recursos comparado con los métodos anteriores.

Extraído de [18] [20] [21] [22]

De los mecanismos de consenso presentados en la Tabla 3, el que se utiliza en el presente proyecto es Practical Byzantine Fault Tolerance (PBFT). Se seleccionó este protocolo de consenso debido a que, es el que está en un desarrollo más maduro de los que ofrece la solución de blockchain usada en el presente proyecto.

2.2.4. Practical Byzantine Fault Tolerance (PBFT)

Este protocolo de consenso se basa en una propiedad de los sistemas que se denomina Byzantine Fault Tolerance (BFT), que es la capacidad de un sistema distribuido de continuar trabajando incluso si algunos nodos fallan o actúan maliciosamente [23].

El protocolo PBFT provee seguridad y continuidad operacional del sistema distribuido siempre y cuando no más de $(n - 1) / 3$ nodos se encuentren en un estado de falla o sean no confiables, siendo n el número total de nodos del sistema [24]. El protocolo PBFT involucra a un nodo denominado cliente, otro nodo denominado primario y dos o más nodos adicionales que son parte del sistema distribuido y trabaja como se muestra a continuación [24]:

1. Una petición parte del nodo cliente y llega a un nodo de la red, el cual se denomina nodo primario.

2. El nodo primario replica la información al resto de nodos.
3. Cada uno de los nodos realiza lo solicitado en la petición y envía la respuesta al cliente.
4. El nodo cliente espera las respuestas de los demás nodos y una vez que reciba un número de respuestas iguales suficientes para lograr la mayoría dentro de la red, se considera a esa respuesta como la válida.

En la Figura 1, se muestra el funcionamiento del algoritmo de PBFT considerando que existe un nodo en estado de falla.

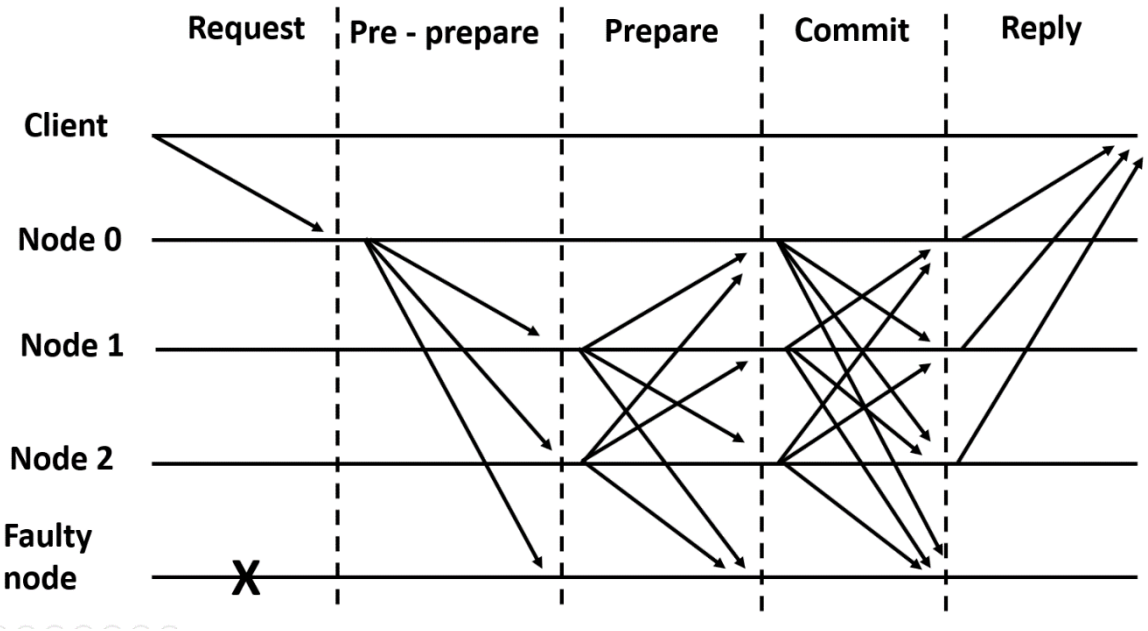


Figura 1 - Algoritmo de PBFT [24]

En las fases de Pre – preparación, Preparación y Commit de la Figura 1, el algoritmo de PBFT realiza las verificaciones correspondientes y la replicación en los nodos de la red, aspecto que no es parte de esta investigación y cuya explicación a detalle se puede encontrar en [24].

Finalmente, “Linux Foundation” con su proyecto Hyperledger, provee soluciones de blockchain de uso libre y, dentro de Hyperledger existe el proyecto Hyperledger Sawtooth, mismo que es la solución que se utilizó en el presente trabajo debido a que es un proyecto maduro que ya ha sido usado en ambientes productivos, como se puede observar en [25].

2.2.5. Hyperledger Sawtooth

Hyperledger Sawtooth es un marco de trabajo para construir DLT's de nivel empresarial. Fue diseñado enfocándose en la seguridad, la escalabilidad y modularidad [26]. Hyperledyer Sawtooth provee un mecanismo para construir una Blockchain de manera que no es necesario desarrollar el código de la cadena de bloques desde cero [27].

Uno de los beneficios de Hyperledger Sawtooth es que permite el despliegue de blockchains privadas y sus principales características se muestran en la Tabla 4.

Tabla 4 - Características de Hyperledger Sawtooth

Especificar el tipo de transacciones que determinados participantes pueden realizar.
Restringir determinados accesos a un determinado grupo de participantes.
Configurar permisos a transacciones basado en políticas definidas.

Extraído de [28]

2.3. Trabajos relacionados

Dentro del área de auditoría de bases de datos, se han presentado trabajos como en [29], en el que se propone un módulo de auditoría embebido en aplicaciones que captura información de auditoría de bases de datos relacionales. En el mencionado trabajo se habla de la necesidad de estos módulos embebidos en aplicaciones para la generación de los datos de auditoría. En el presente trabajo los datos se generan desde la misma base de datos, lo que lo hace una solución más desacoplada ya que no es necesario alterar el funcionamiento de las aplicaciones y más bien, cualquier aplicación que realice alguna alteración de los datos de la base generará datos de auditoría.

Existen muchos trabajos relacionados con la seguridad en base de datos, sin embargo, enlazar la seguridad de una base de datos con la tecnología de blockchain es un ámbito que no ha sido explorado lo suficiente debido principalmente a que se tiene la idea de que blockchain tiene falencias en cuanto a su rendimiento. Esto ya ha venido mejorando desde hace algunos años con proyectos que implementan blockchain de manera exitosa en ambientes productivos sin presentar inconvenientes en cuanto al rendimiento como se evidencia en [25].

En bases de datos distribuidas uno de los grandes desafíos es la consolidación de eventos ya que al tener transacciones provenientes de varios nodos o servidores, se

podría tener transacciones que aparentemente se registraron al mismo tiempo, pero para temas de auditoría es necesario saber que transacción se realizó primero, este apartado se estudia en [30], en donde se explica un sistema de replicación y se realiza el control de la concurrencia y la consolidación de los datos mediante bloqueos temporales de las transacciones. En el presente trabajo, el manejo de la consolidación es llevada a cabo por las funciones de la blockchain.

En cuanto a blockchain, en los últimos años se han escrito muchos estudios que utilizan los beneficios de esta tecnología. En [31] se presenta una evaluación del rendimiento de Hyperledger Sawtooth y muestra que es una solución que puede ser utilizada en entornos empresariales. En [32] Hyperledger Sawtooth se utiliza en Internet de las cosas para generar una plataforma de seguimiento de la ubicación de medicamentos. En [33] los autores proponen un método para validar datos de auditoría usando contratos inteligentes de Ethereum [34]. En [35] se presenta un ecosistema de blockchains interconectados con una blockchain externa mediante procedimientos de auditoría inteligentes, dicho trabajo se enfoca en registrar información financiera. El presente trabajo se centra en registrar la información de cualquier operación DML en bases de datos distribuidas. Una solución similar a la presentada en este artículo se expone en [13] utilizando un mecanismo con relojes lógicos y procedimientos almacenados para registrar y generar una línea de tiempo de operaciones DML en una base de datos.

La red de firmas de servicios de auditoría, asesoramiento legal y fiscal, y asesoramiento financiero y de negocio (KPMG), en su encuesta realizada entre diciembre de 2018 y enero de 2019 a más de 740 líderes en la industria tecnológica en 20 países acerca de la innovación tecnológica en la industria, indica que el 41% de los encuestados respondieron que probablemente sus compañías implementen tecnología blockchain entre 2020 y 2021 [36]. Además, en el mismo estudio, el 48% de los encuestados respondió que probablemente blockchain cambiaría la forma en que sus compañías realizan sus operaciones entre 2020 y 2021 [36].

3. IMPLEMENTACIÓN

En esta sección, se describe como se llevó a cabo la implementación de la prueba de concepto realizada sobre la arquitectura que se propone.

3.1. Diseño de la solución

Según la metodología seleccionada, se debe producir un artefacto, en este caso, es un modelo de una arquitectura que integra una base de datos con una blockchain para secuenciar los eventos de una base de datos y determinar si es factible usar este tipo de arquitectura para el fin que se busca, para ello, se llevó a cabo una prueba de concepto realizando las actividades que se detallan a continuación:

3.1.1. Elección de la tecnología para la base de datos

Para la solución propuesta, se requiere que la base de datos se comunique con una aplicación externa como lo es la blockchain, se consideraron gestores de base de datos relacionales cómo: MySQL, MariaDB, PostgresQL, Oracle y Microsoft SQL Server, de las cuales, la que provee un mecanismo relativamente fácil de implementar para permitir la comunicación con aplicaciones externas es Microsoft SQL Server con su servicio “Service Broker”. Además, esta base de datos es una de las más utilizadas a nivel empresarial, por lo que es ideal para realizar la implementación de la arquitectura propuesta.

3.1.2. Elección de la tecnología para la blockchain

Actualmente existen varios proyectos que implementan blockchain, como Hyperledger Fabric, Hyperledger Iroha o Hyperledger Sawtooth [37], que son parte de “Linux Foundation” y el proyecto BigchainDB [38]. Se consideraron estos proyectos ya que son libres y no conllevan costos de licenciamiento adicionales. Además, se decidió utilizar una de las soluciones de blockchain que existen actualmente en el mercado para usos empresariales y no desarrollar una propia debido a que no es el objetivo principal del presente proyecto y al seleccionar una de estas opciones, queda en segundo plano los problemas de la replicación de la información y del mecanismo de consenso, ya que estos ya están cubiertos por las soluciones existentes. De las soluciones expuestas, se decidió utilizar Hyperledger Sawtooth, ya que es uno de los proyectos más maduros de Linux Foundation, permite agregar funcionalidades personalizadas a la blockchain y es más flexible en cuanto s du funcionamiento, por ejemplo, varias de los proyectos mencionados se enfocan únicamente al ámbito de las criptomonedas y transacciones.

3.1.3. Esquema de la solución

Se ha mencionado que, para cubrir los requerimientos de auditoría, es necesario que se completen las etapas de: **generación**, **recolección** y **conservación** de pistas de

auditoría; tales etapas, en el presente proyecto, se llevan a cabo mediante el esquema de la Figura 2

Al realizar el proceso de generación, recolección y conservación simultáneamente en varios servidores de una base de datos distribuida, pero conectados a una única blockchain, se genera en la propia blockchain una línea de tiempo global de las operaciones realizadas sobre la base de datos que incluye a todos los servidores de la base de datos.

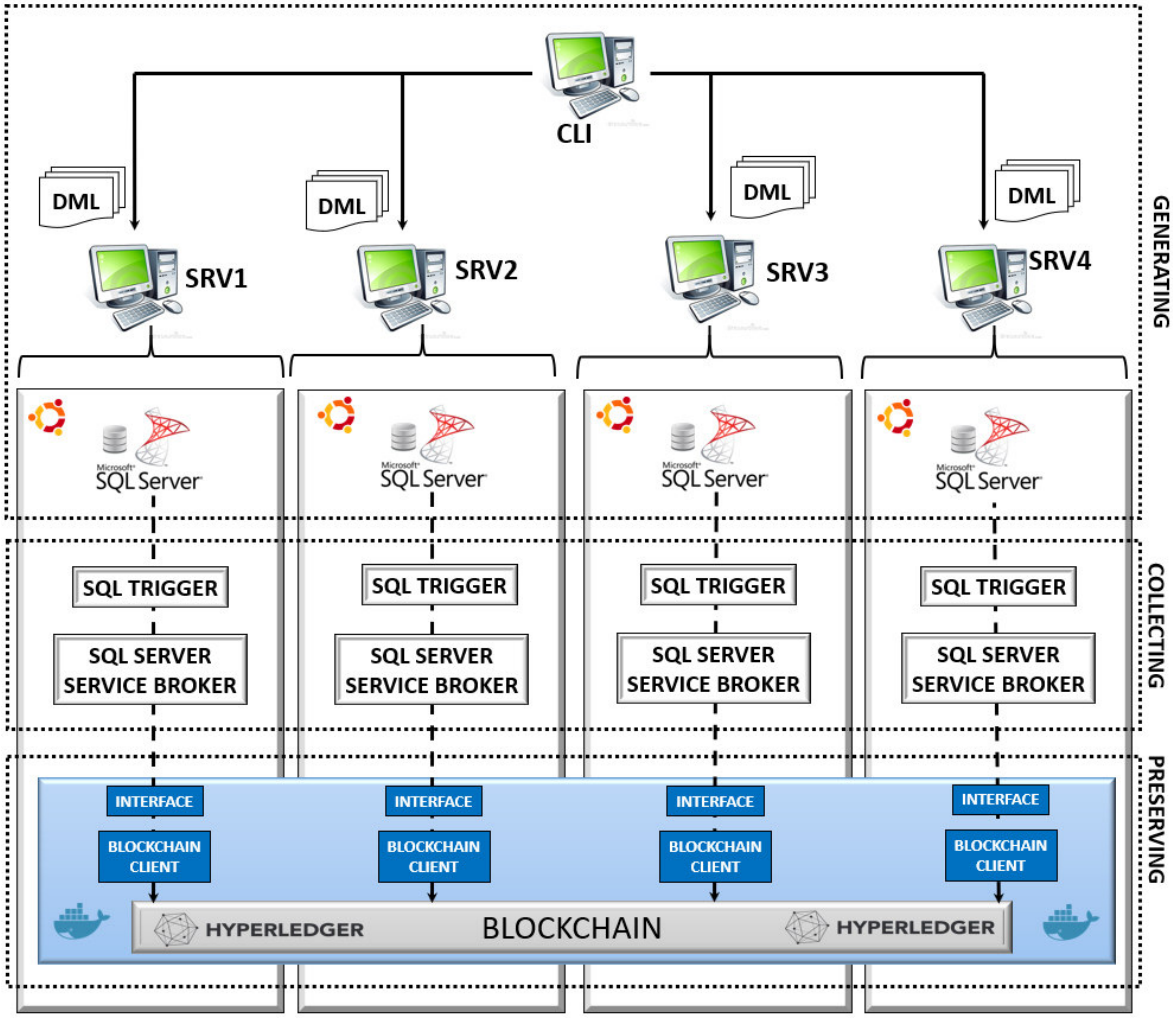


Figura 2 - Esquema de la solución

Para evaluar el esquema de la Figura 2, fueron creadas 5 máquinas virtuales en Google Cloud Platform (GCP) Compute Engine [39] con las características indicadas en la Tabla 5. Cuatro máquinas virtuales para poner en marcha la blockchain y una máquina virtual con el software Apache JMeter instalado para lanzar las pruebas. Se decidió configurar 4 servidores para la blockchain porque la solución de blockchain elegida con el protocolo

de consenso seleccionado así lo sugiere como número mínimo de nodos para su correcto funcionamiento.

Tabla 5 - Características de las máquinas virtuales de GCP

Etiqueta	**Tipo**	**Sistema Operativo**	**Rol**	**CPU's**	**RAM**	**Disco**	**IP Interna**
SRV1	e2-standard-4	Ubuntu Server 20.04	Nodo Blockchain 1	4 CPU's virtuales dedicados	16 GB	10 GB	10.142.0.2
SRV2	e2-standard-4	Ubuntu Server 20.04	Nodo Blockchain 2	4 CPU's virtuales dedicados	16 GB	10 GB	10.142.0.3
SRV3	e2-standard-4	Ubuntu Server 20.04	Nodo Blockchain 3	4 CPU's virtuales dedicados	16 GB	10 GB	10.142.0.4
SRV4	e2-standard-4	Ubuntu Server 20.04	Nodo Blockchain 4	4 CPU's virtuales dedicados	16 GB	10 GB	10.142.0.5
CLI	e2-medium	Windows Server 2019 Datacenter	Cliente con Apache JMeter	2 CPU's virtuales compartidos	4 GB	50 GB	10.142.0.9

Al completar el proceso de generación, recolección y preservación simultáneamente en varios servidores distribuidos pero conectados a una única blockchain, se genera en esta una línea de tiempo global de las transacciones realizadas sobre la base de datos que incluye a todos los servidores distribuidos.

En la arquitectura experimental se usó Microsoft SQL Server como gestor de base de datos con la característica "Service Broker" habilitada. SQL Server Service Broker es una arquitectura que permite escribir aplicaciones desacopladas, distribuidas, persistentes, confiables, escalables y seguras basadas en colas o mensajes asincrónicos dentro de la propia base de datos [40]. Gracias a Service Broker no es necesario leer constantemente las tablas y realizar algún proceso durante el ciclo de vida de la consulta. Una aplicación que usa Service Broker es llamada "aplicación orientada al servicio" y, en este tipo de aplicaciones, la base de datos exporta servicios que permanentemente intercambian mensajes (datos de una transacción). Cada uno de estos servicios maneja una cola en la

que los mensajes son depositados hasta que sean procesados [41]. En el caso de la solución propuesta en este trabajo, el proceso termina cuando los datos de la transacción son registrados en la blockchain.

En la Figura 3 se muestra un script de ejemplo para habilitar Service Broker en una base de datos de SQL Server. El script permite a la base de datos generar una cola para enviar mensajes a una aplicación externa, en este caso, a una interfaz que interactúa con la blockchain. El Anexo I muestra la guía técnica para la implementación de la solución.

```
CREATE MESSAGE TYPE
    [http://audit_blockchail/RequestMessage]
VALIDATION = WELL_FORMED_XML
CREATE MESSAGE TYPE
    [http://audit_blockchail/ResponseMessage]
VALIDATION = WELL_FORMED_XML
CREATE CONTRACT [http://audit_blockchail/Contract]
(
[http://audit_blockchail/RequestMessage] SENT BY
    INITIATOR,
[http://audit_blockchail/ResponseMessage] SENT BY
    TARGET
)
CREATE QUEUE InitiatorQueue
WITH STATUS = ON
CREATE QUEUE TargetQueue
CREATE SERVICE InitiatorService
ON QUEUE InitiatorQueue
(
[http://audit_blockchail/Contract]
)
CREATE SERVICE TargetService
ON QUEUE TargetQueue
(
[http://audit_blockchail/Contract]
)
CREATE QUEUE ExternalActivatorQueue
CREATE SERVICE ExternalActivatorService
ON QUEUE ExternalActivatorQueue
(
http://schemas.microsoft.com/SQL/Notifi
cations/PostEventNotification]
)
CREATE EVENT NOTIFICATION
    EventNotificationTargetQueue
ON QUEUE TargetQueue
FOR QUEUE_ACTIVATION
TO SERVICE 'ExternalActivatorService', 'current
    database';
```

Figura 3 - Script ejemplo para habilitar SQL Server Service Broker

3.1.4. Fase de generación

La generación de los datos de auditoría se realiza cada vez que desde alguna aplicación externa o desde la propia base de datos se ejecuta un INSERT, UPDATE o DELETE en alguna de las tablas consideradas para generar los registros de auditoría.

Para testear la arquitectura experimental, se usó el software Apache JMeter para generar y enviar los datos hacia la base de datos, la fase de generación sigue el flujo del algoritmo de la Figura 4.

Algorithm 1: Generating transaction data - Apache JMeter (SQL Request)

```
01: SET DATABASE SESSION VARIABLES
02: {
03: application_time = GET CURRENT DATETIME;
04: application_user = GET APPLICATION USER;
05: }
06: data = GENERATE TEST DATA;
07: EXECUTE INSERT UPDATE OR DELETE (data);
```

Figura 4 – Algoritmo de generación

En el lado del cliente (Apache JMeter), se configuró un controlador de orden aleatorio (Random Order Controller) y 4 elementos de petición JDBC (cada una apuntando a cada uno de los servidores) para simular transacciones simultaneas aleatoriamente en todos los servidores. En cada petición JDBC, se crean 2 variables de sesión en la base de datos (Líneas 2 y 3 del algoritmo de generación) y una transacción de inserción, actualización o eliminación con datos aleatorios. Luego, en la fase de recolección, estas dos variables serán leídas y se agregarán datos de auditoría adicionales para ser enviados a la cola de Service Broker.

Las líneas 2 y 3 del algoritmo de la Figura 4 se pueden realizar en SQL Server mediante el comando:

```
EXEC sys.sp_set_session_context @key = N'session_variable_name', @value = 'session_variable_value'
```

Estas variables de sesión serán leídas desde un trigger en la fase de recolección.

Cabe mencionar que, para el funcionamiento de la arquitectura en ambientes de producción, no sería estrictamente necesario generar estas 2 variables ya que implicaría modificar el funcionamiento de las aplicaciones que se conectan a la base de datos. En este caso se lo realiza de esta manera para poder comprobar que la arquitectura conserva el orden de las transacciones.

3.1.5. Fase de recolección

Cuando se realiza un INSERT, UPDATE o DELETE sobre alguna de las tablas consideradas, se ejecuta un TRIGGER que envía los datos de la transacción a una cola de Service Broker, luego, mediante una interfaz desarrollada como parte de la solución propuesta, los datos de la transacción son leídos de la cola de Service Broker y agregados a un bloque de la blockchain con información adicional como el tipo de transacción (INSERT, UPDATE o DELETE), la tabla afectada, el servidor desde el cual se realizó la transacción, el usuario tanto de aplicación como de base de datos que ejecutó la transacción y los tiempos en los que la información llegó a la base de datos y a la blockchain. Esta fase sigue el flujo del algoritmo de la Figura 5.

Y el script de la Figura 6 muestra un ejemplo de un trigger para recolectar los datos de auditoría. Estos datos serán procesados y almacenados en la blockchain luego en la fase de preservación.

Algorithm 2: Collecting transaction and audit data - SQL Trigger

```
01: READ INSERT UPDATE OR DELETE DATA
FROM EXECUTED QUERY;
02: READ DATABASE SESSION VARIABLES
03: {
04: application_time;
05: application_user;
06: }
07: SET VARIABLES
08: {
09: database_time = GET CURRENT DATETIME;
10: database_name = GET DATABASE NAME;
11: table_name = GET TABLE NAME;
12: transaction = INSERT UPDATE OR DELETE;
13: database_user = GET DATABASE USER;
14: database_host = GET DATABASE HOST NAME;
15: }
16: SEND COLLECTED DATA TO SERVICE
 BROKER CONFIGURED QUEUE;
```

Figura 5 – Algoritmo de recolección

```
1  CREATE TRIGGER dbo.tr_insert_table_name
2  ON dbo.table_name FOR INSERT AS
3  BEGIN
4  DECLARE
5  @ch UNIQUEIDENTIFIER,
6  @data NVARCHAR(4000),
7  @database_time DATETIME;
8  BEGIN TRANSACTION;
9  BEGIN DIALOG CONVERSATION @ch
10 FROM SERVICE [InitiatorService]
11 TO SERVICE 'TargetService'
12 ON CONTRACT [http://audit_blockchail/Contract]
13 WITH ENCRYPTION = OFF;
14 SELECT @database_time = GETDATE();
15 SET @data = (SELECT 'database_name' AS
       '___database___',
16 'table_name' AS '___table___',
17 'INSERT' AS '___transaction___',
18 REPLACE(SYSTEM_USER,'\','_') AS
       '___database_user___',
19 @database_time AS '___database_time___',
20 SESSION_CONTEXT(N'application_time') AS
       '___application_time___',
21 SESSION_CONTEXT(N'application_user') AS
       '___application_user___',
22 HOST_NAME() AS '___client_host___',
23 @@SERVERNAME AS '___database_host___',
24 * FROM inserted FOR XML AUTO, ELEMENTS);
25 ;SEND ON CONVERSATION @ch
26 MESSAGE TYPE
       [http://audit_blockchail/RequestMessage]
       (@data);
27 COMMIT;
28 END
29 GO
```

Figura 6 - Script ejemplo del trigger de recolección de datos de auditoría

En el algoritmo de la Figura 5, la línea 1 corresponde a los datos del objeto “inserted” y / o “deleted” dependiendo de la transacción realizada sobre la base de datos. Cuando se ejecuta una inserción, se crea un objeto “inserted”, que puede ser leído desde un trigger, con los valores del nuevo registro. Cuando se ejecuta una actualización, se crean los objetos “deleted” e “inserted” con los valores de registro antes de la actualización y después de la actualización respectivamente, los valores de ambos objetos son almacenados en la blockchain para poder tener información de cómo estaba dicho registro originalmente si resulta que la actualización no fuese legítima. Y, cuando se ejecuta una eliminación, el objeto “deleted” se crea con los valores del registro eliminado y puede ser leído desde el trigger para ser registrado en la blockchain.

De las líneas 9 a 14 del algoritmo de la Figura 5, recolectan información adicional de auditoría y en la línea 16, todos los datos recolectados son enviados a una cola de Service Broker, para que luego sean leídos en la fase de preservación mediante la interfaz desarrollada como parte de la solución. En dicha fase, estos datos serán enviados a la blockchain para completar el proceso.

La línea 2 del algoritmo de la Figura 5, se puede llevar a cabo mediante el comando:

SESSION_CONTEXT(N’session_variable_name’)

3.1.6. Fase de preservación

Una vez que los datos son registrados en la blockchain, la ocurrencia de la operación realizada en la base de datos quedará permanentemente registrada en la cadena de bloques asegurando la conservación de las pistas de auditoría. Es importante mencionar que la blockchain se ejecuta sobre contenedores Docker, lo que hace que la solución propuesta pueda ser implementada en cualquier plataforma que soporte esta tecnología. Esta fase sigue el flujo del algoritmo de la Figura 7. Para completar esta fase, un componente interfaz y un componente cliente de blockchain fueron desarrollados usando NodeJS.

La comunicación entre la cola de SQL Server Service Broker y el cliente de blockchain desarrollado, se realiza mediante el código de la Figura 8, este código es parte de la interfaz. Y la comunicación entre el cliente blockchain y la propia blockchain se realiza mediante el código de la Figura 9.

Tanto la interfaz como el cliente blockchain usan métodos HTTP para enviar mensajes y completar la interacción entre la base de datos y la cadena de bloques.

Algorithm 3: Preserving transaction and audit data - Blockchain

```
01: WHILE (IS DATA IN SQL SERVICE BROKER QUEUE)
02: READ NEXT SQL SERVICE BROKER QUEUE REGISTRY;
03: TRANSFORM QUEUED REGISTRY TO JSON FORMAT;
04: ADD BLOCKCHAIN TIME TO DATA;
05: SEND DATA TO DEVELOPED BLOCKCHAIN CLIENT;
06: ADD BLOCKCHAIN HOST NAME TO DATA;
07: MAKE A POST REQUEST TO BLOCKCHAIN REST-API;
08: END WHILE
```

Figura 7 - Algoritmo de preservación

```
servicel.on("http://audit_blockchail/
RequestMessage", async ctx -> {
let res - await fetch("http://client/saveAudit/",
    {
method: "post",
body: JSON.stringify(xmlToJson
(ctx.messageBody)),
headers: { "Content-Type": "application/json" }
});
await res.json();
});
```

Figura 8 – Comunicación entre Service Broker y el cliente blockchain

```
let res - await
    fetch("http://blockchain-rest-api:8008/
batches", {
method: "post",
body: batchListBytes,
headers: { "Content-Type":
    "application/octet-stream" }
});
await res.json();
```

Figura 9 - Comunicación entre el cliente blockchain y la blockchain

Al completar la fase de preservación, quedan cubiertos los requerimientos de auditoría y se genera en la blockchain un registro con datos de auditoría de las transacciones realizadas sobre la base de datos que preserva la secuencia de estas.

Finalmente, como parte de la solución propuesta, se provee un tablero de Grafana [42] para monitorear el rendimiento de la propia blockchain. En este tablero se pueden ver métricas como el número de bloques y transacciones registradas. En el Anexo II se muestran los pasos para configurar el tablero de Grafana adaptado a la solución.

3.2. Evaluación de la solución

Como ya se ha mencionado, para la evaluación se usó el software Apache JMeter, este software permite ejecutar conjuntos de pruebas automáticamente. Para el caso específicos de pruebas para bases de datos, en cada uno de los mencionados conjuntos se puede ejecutar un script SQL un número determinado de veces en un tiempo especificado, dentro de cada script se puede incluir inserciones, actualizaciones o eliminaciones además de cualquier otra instrucción SQL. La solución fue testeada ejecutando 3 conjuntos de inserciones, 3 conjuntos de actualizaciones y 3 conjuntos de eliminaciones considerando un nivel de carga bajo, medio y alto. Se consideró realizar 3 conjuntos para asegurar que los resultados sean estables y eliminar cualquier desviación que pueda presentarse por factores como intermitencia en la red. Durante las pruebas se evidenció que, al llegar al tercer conjunto los valores medidos ya eran estables. Para el nivel bajo se consideraron 100 transacciones en un segundo, para el nivel bajo 500 transacciones en un segundo y para el nivel alto 1000 transacciones en un segundo por cada conjunto. Se determinó que sean 1000 transacciones en un segundo como nivel máximo debido a que, con las características de hardware definidas para las pruebas, se presentaron problemas de disponibilidad en los servidores para cargas mayores.

Luego, se realizaron las mismas pruebas con la blockchain y triggers desactivados para comparar los resultados del rendimiento y determinar la factibilidad de la solución.

Se presentan dos tipos de evaluación: de rendimiento para determinar la aplicabilidad de la solución (siempre considerando que con servidores con mejores prestaciones de hardware se obtendrían mejores resultados); y de consistencia de tiempo con el fin de saber si la solución conserva el orden correcto de las transacciones considerando todos los servidores de la base de datos.

3.2.1. Evaluación de rendimiento

Las figuras presentadas en esta sección fueron extraídas del Monitor de Recursos de Google Cloud Platform, en estas, cada color representa cada nodo de la blockchain, sin

embargo, para propósitos del análisis de rendimiento no resulta relevante analizar cada uno de los nodos y más bien, se analiza el comportamiento global.

A. Cien transacciones

La Figura 10 - A muestra el uso de procesador luego de ejecutar 3 conjuntos de 100 inserciones, 3 conjuntos de 100 actualizaciones y 3 conjuntos de 100 eliminaciones (900 transacciones en total) con la blockchain activada y la Figura 10 - B muestra el uso de procesador para el mismo caso con la blockchain desactivada.

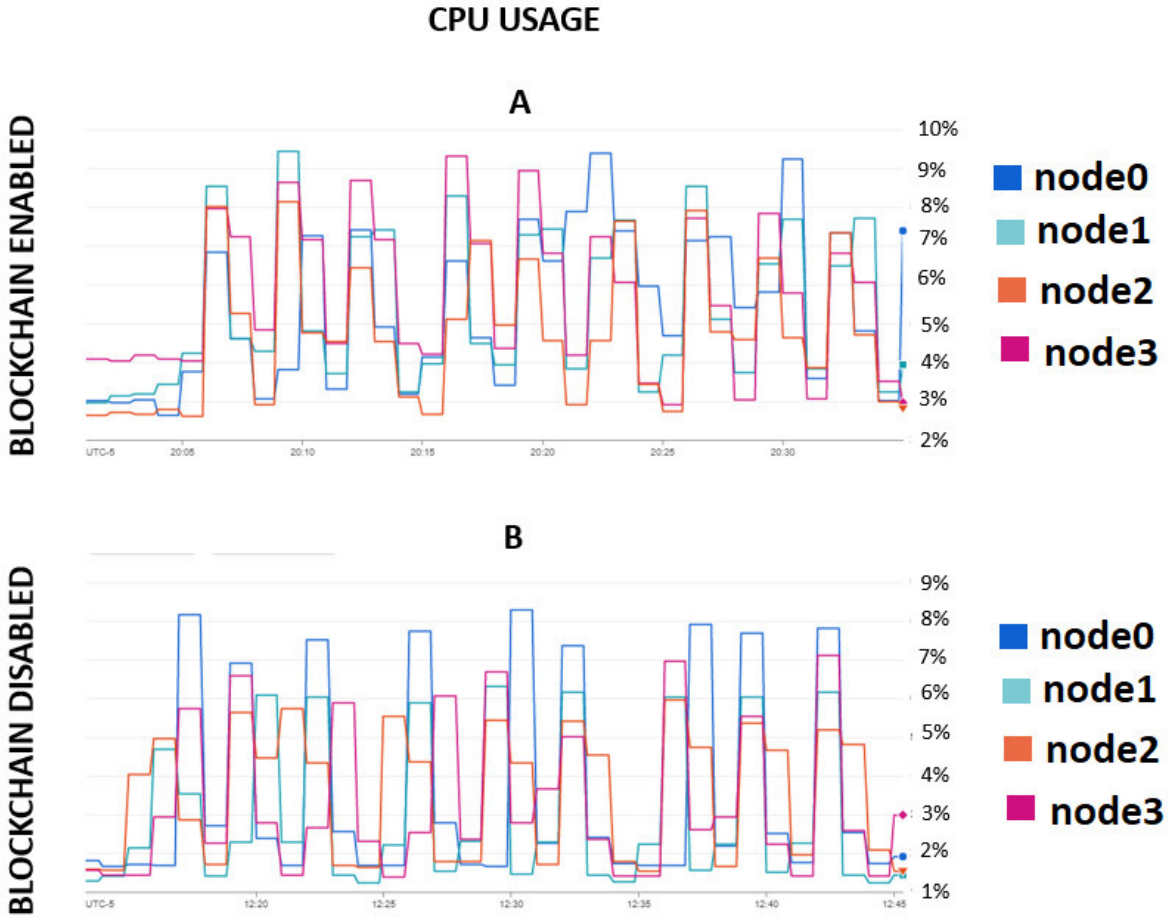


Figura 10 – Uso de procesador con 100 transacciones

Al ejecutar 100 transacciones, no parece haber una diferencia significativa en el uso de procesador entre tener la blockchain activa o inactiva. Es decir, en ambientes con poco volumen de transacciones, no se vería diferencias en el uso de procesador con la implementación de la solución propuesta.

La Figura 11 - A muestra los bytes transmitidos en la red luego de ejecutar 3 conjuntos de 100 inserciones, 3 conjuntos de 100 actualizaciones y 3 comjuntos de 100 eliminaciones con la blockchain activada y la Figura 11 - B muestra los bytes de red transmitidos para el mismo caso con la blockchain desactivada.

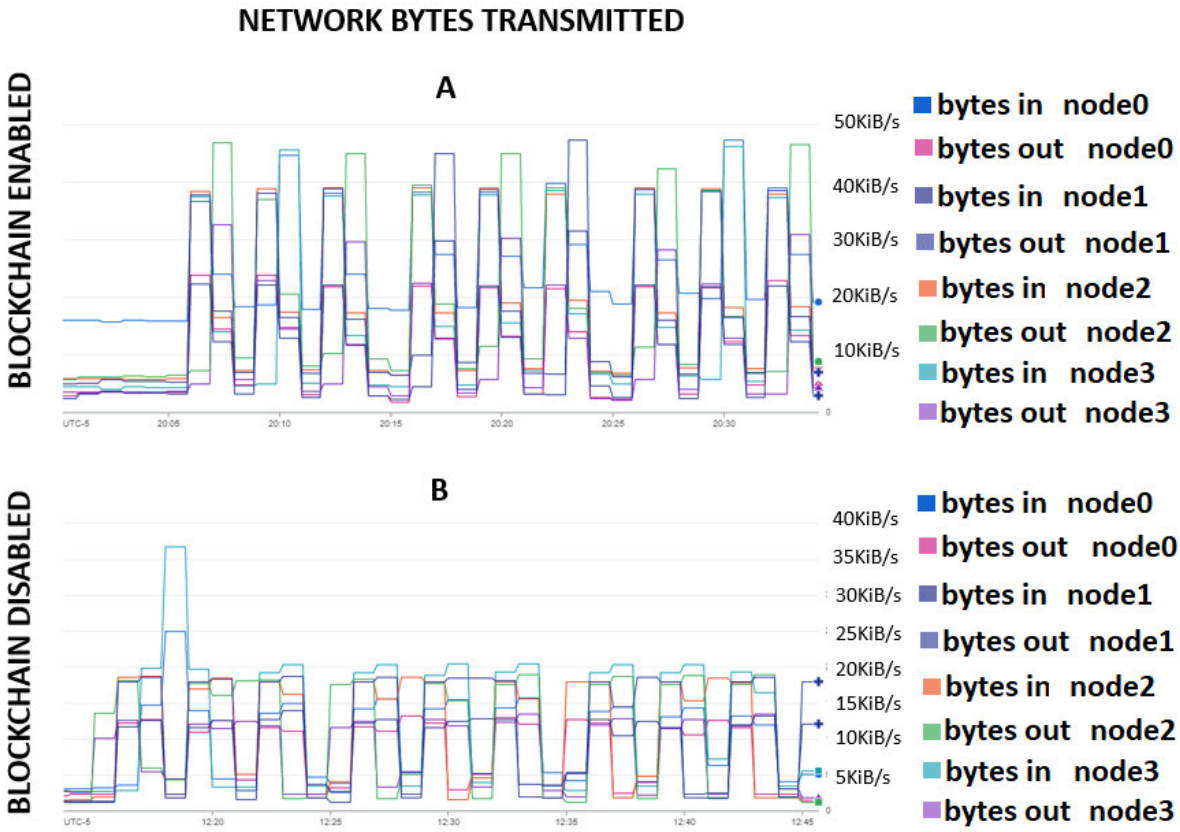


Figura 11 - Bytes de red transmitidos con 100 transacciones

En cuanto a los bytes transmitidos en la red para 100 transacciones, existe una pequeña diferencia de aproximadamente 20 KiB/s, que sigue siendo un valor totalmente aceptable para ambientes productivos.

La Figura 12 - A muestra las operaciones de Lectura / Escritura en disco luego de ejecutar 3 conjuntos de 100 inserciones, 3 conjuntos de 100 actualizaciones y 3 conjuntos de 100 eliminaciones con la blockchain activada y la Figura 12 - B muestra los bytes de red transmitidos para el mismo caso con la blockchain desactivada.

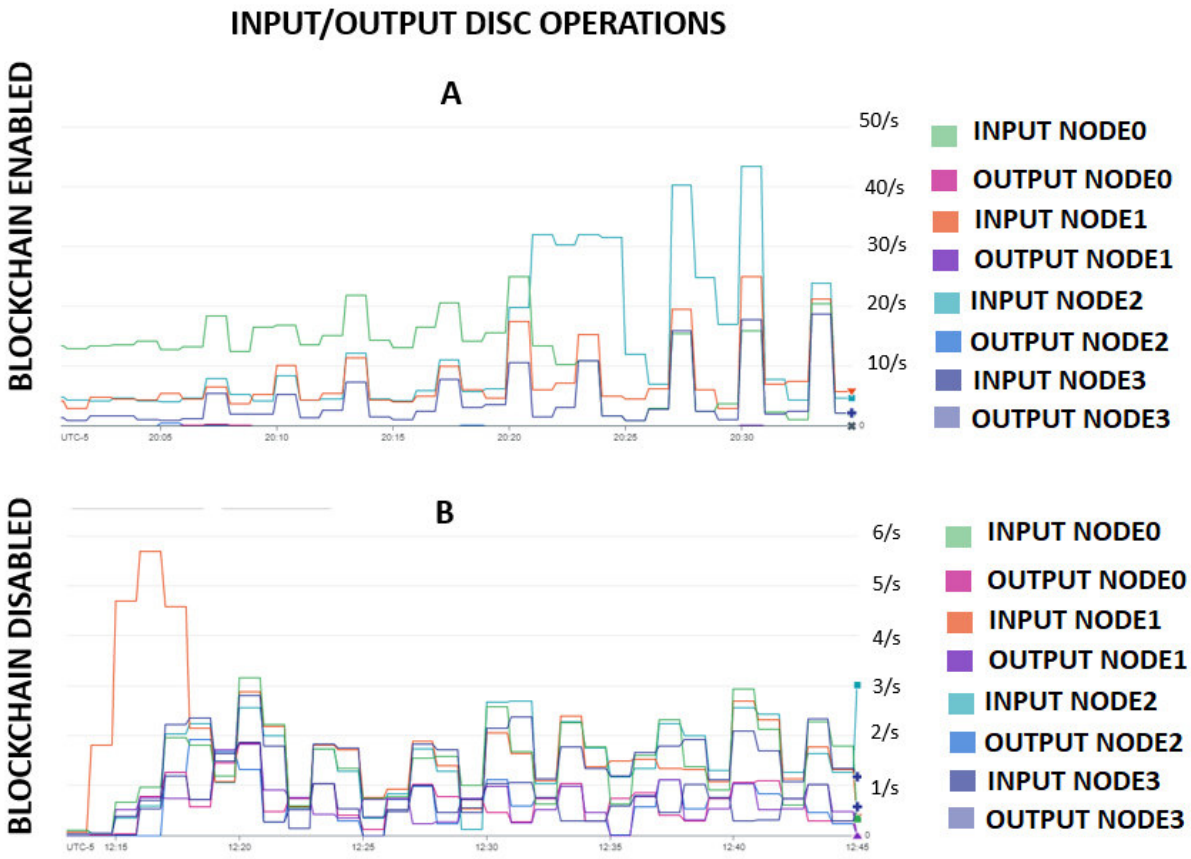


Figura 12 - Operaciones de Lectura / Escritura en disco con 100 transacciones

Las operaciones de lectura / escritura de disco para 100 transacciones se multiplican aproximadamente por 10 con la blockchain activa. Lo que implica que el disco podría convertirse en un componente de suma importancia para la implementación de la solución propuesta.

B. Quinientas transacciones

La Figura 13 - A muestra el uso de procesador luego de ejecutar 3 conjuntos de 500 inserciones, 3 conjuntos de 500 actualizaciones y 3 conjuntos de 500 eliminaciones (4500 transacciones en total) con la blockchain activada y la Figura 13 - B muestra el uso de procesador para el mismo caso con la blockchain desactivada.

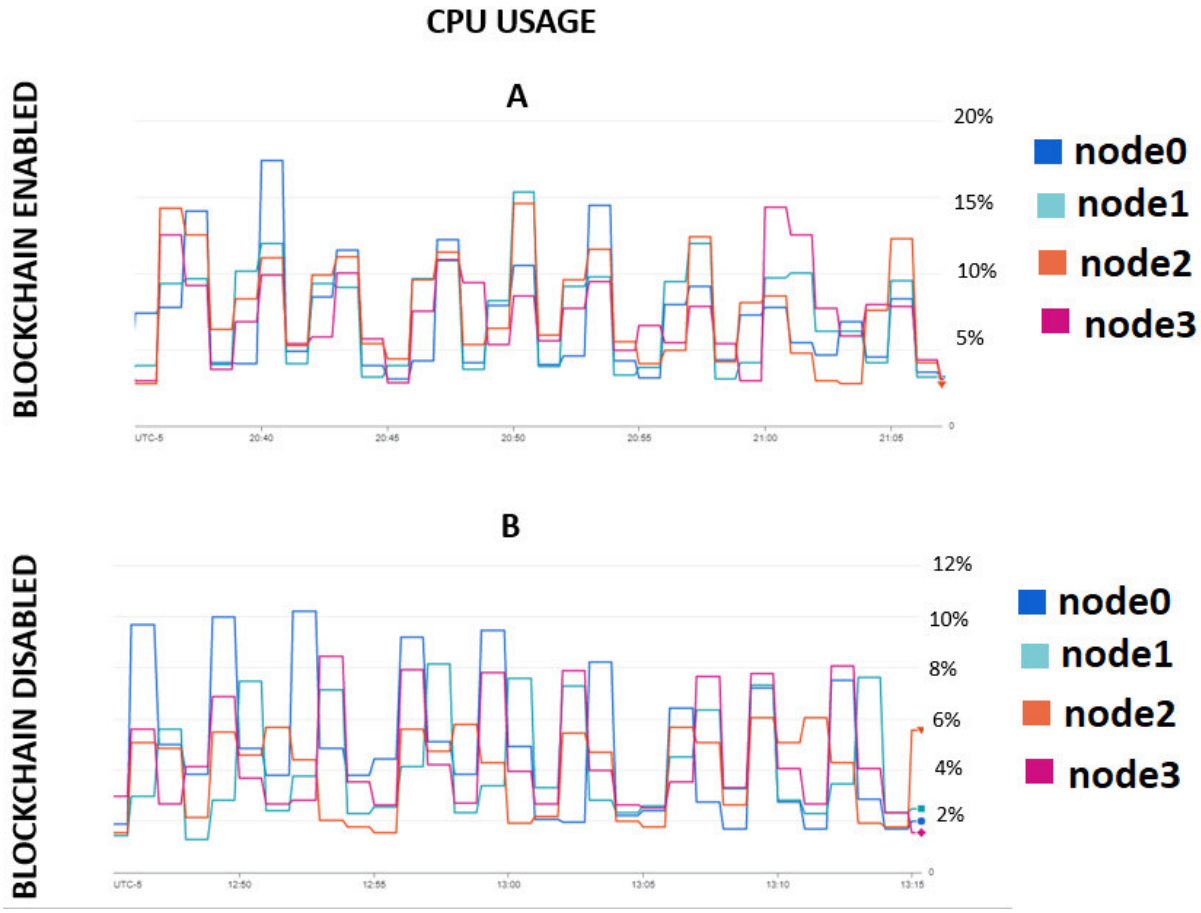


Figura 13 - Uso de procesador con 500 transacciones

Al ejecutar 500 transacciones, el uso de procesador se aumenta aproximadamente en un 5% con la blockchain activa, lo que sigue siendo un incremento bajo y factible para implementaciones en ambientes de producción.

La Figura 14 - A muestra los bytes transmitidos en la red luego de ejecutar 3 conjuntos de 500 inserciones, 3 conjuntos de 500 actualizaciones y 3 conjuntos de 500 eliminaciones con la blockchain activada y la Figura 14 - B muestra los bytes de red transmitidos para el mismo caso con la blockchain desactivada.

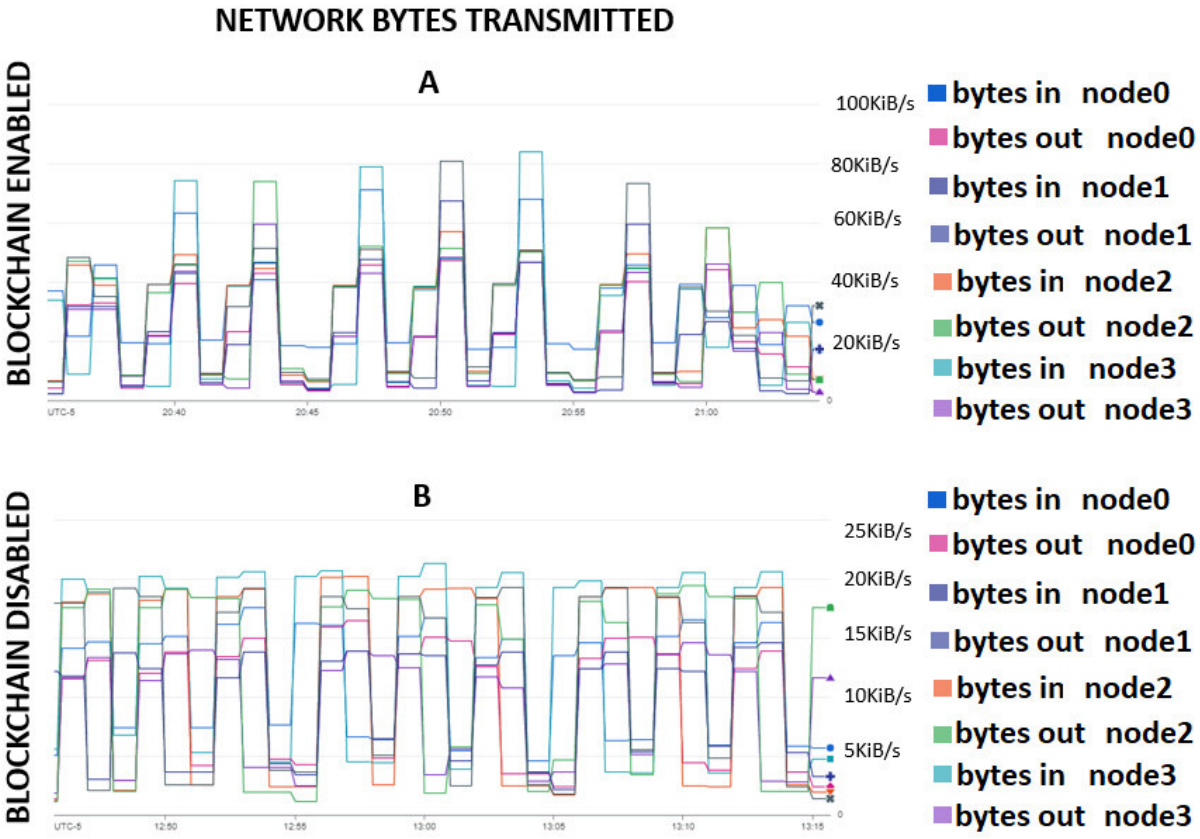


Figura 14 - Bytes de red transmitidos con 500 transacciones

Los bytes transmitidos en la red con 500 transacciones se multiplican aproximadamente por 4 con la blockchain activa.

La Figura 15 - A muestra las operaciones de Lectura / Escritura en disco luego de ejecutar 3 conjuntos de 500 inserciones, 3 conjuntos de 500 actualizaciones y 3 conjuntos de 500 eliminaciones con la blockchain activada y la Figura 15 - B muestra los bytes de red transmitidos para el mismo caso con la blockchain desactivada.

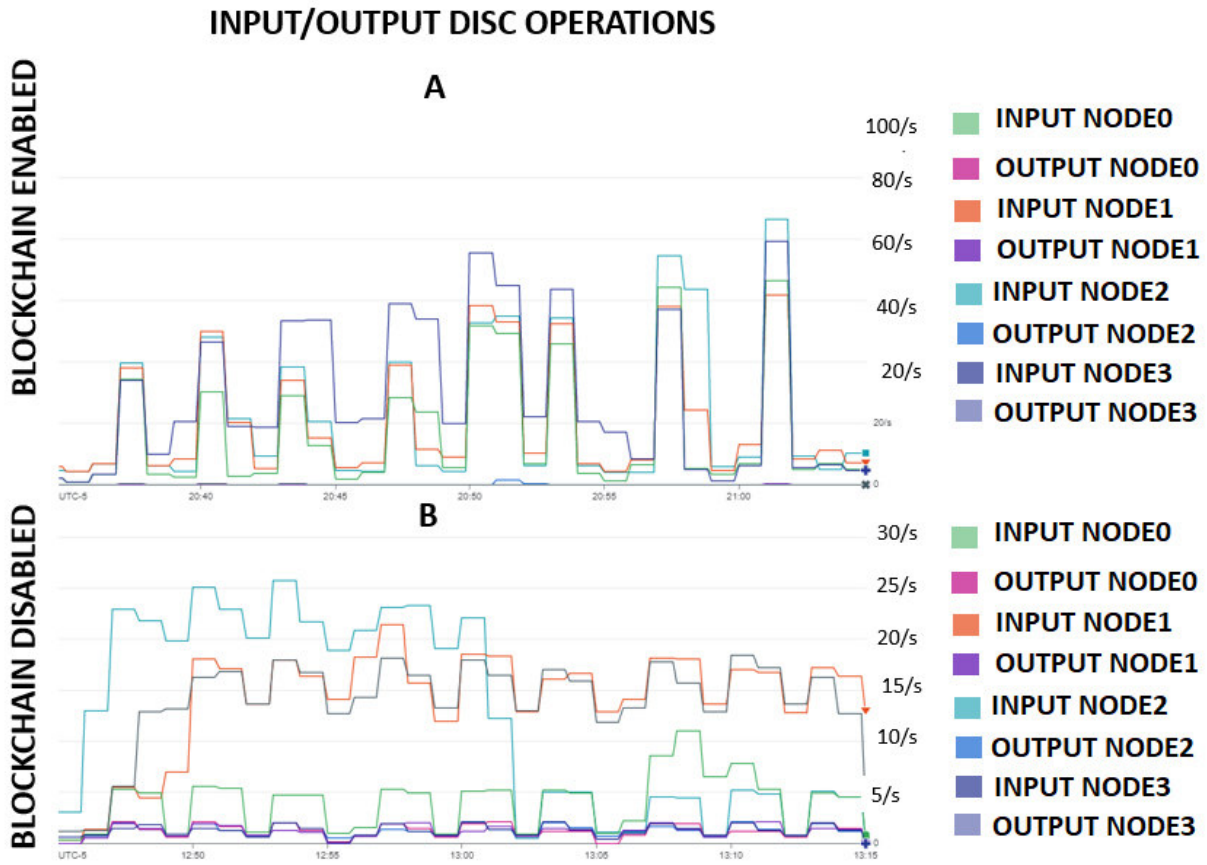


Figura 15 - Operaciones de Lectura / Escritura en disco con 500 transacciones

Al igual que con los bytes transmitidos, las operaciones de lectura / escritura de disco se multiplican aproximadamente por 4 con la blockchain activa.

C. Mil transacciones

La Figura 16 - A muestra el uso de procesador luego de ejecutar 3 conjuntos de 1000 inserciones, 3 conjuntos de 1000 actualizaciones y 3 conjuntos de 1000 eliminaciones (9000 transacciones en total) con la blockchain activada y la Figura 16 - B muestra el uso de procesador para el mismo caso con la blockchain desactivada.

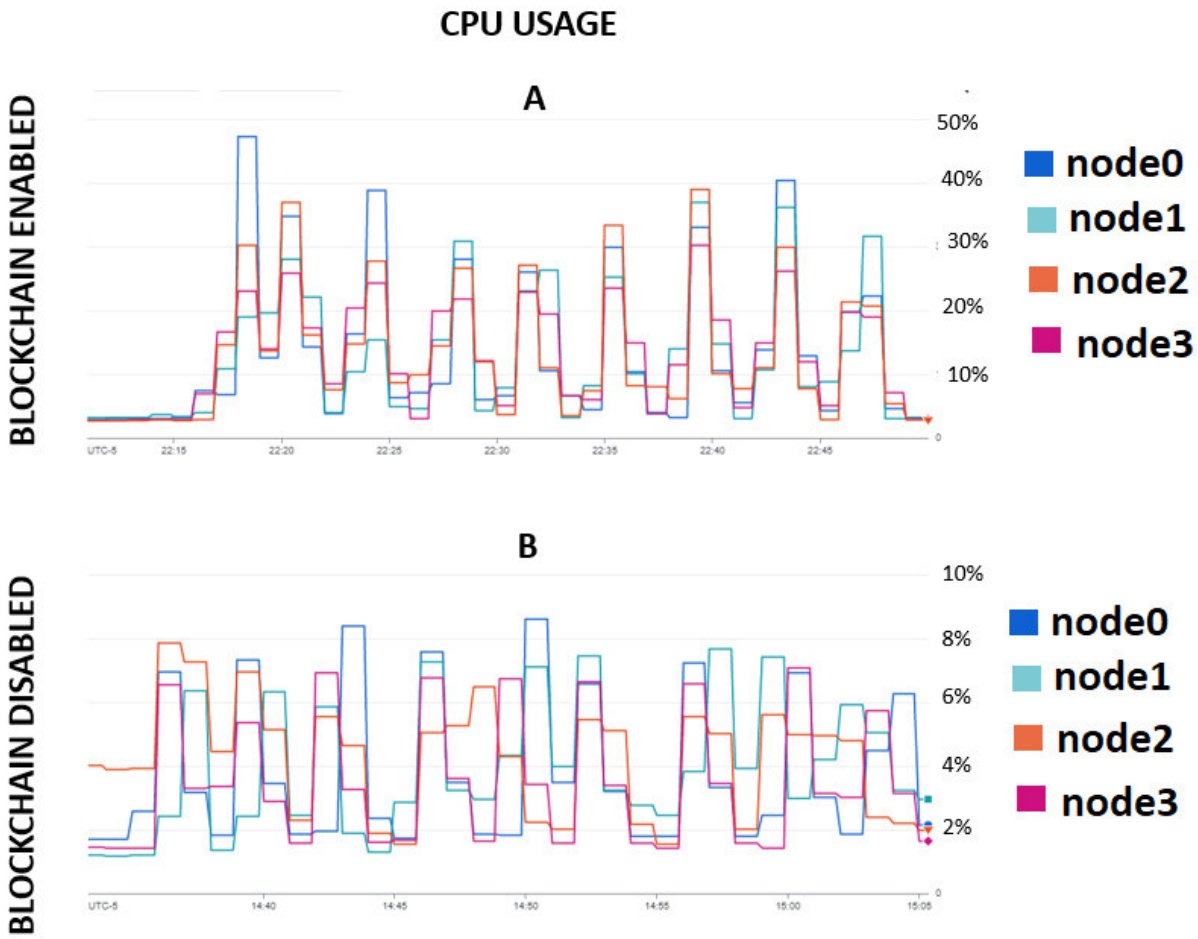


Figura 16 - Uso de procesador con 1000 transacciones

La Figura 17 - A muestra los bytes transmitidos en la red luego de ejecutar 3 conjuntos de 1000 inserciones, 3 conjuntos de 1000 actualizaciones y 3 conjuntos de 1000 eliminaciones con la blockchain activada y la Figura 17 - B muestra los bytes de red transmitidos para el mismo caso con la blockchain desactivada.

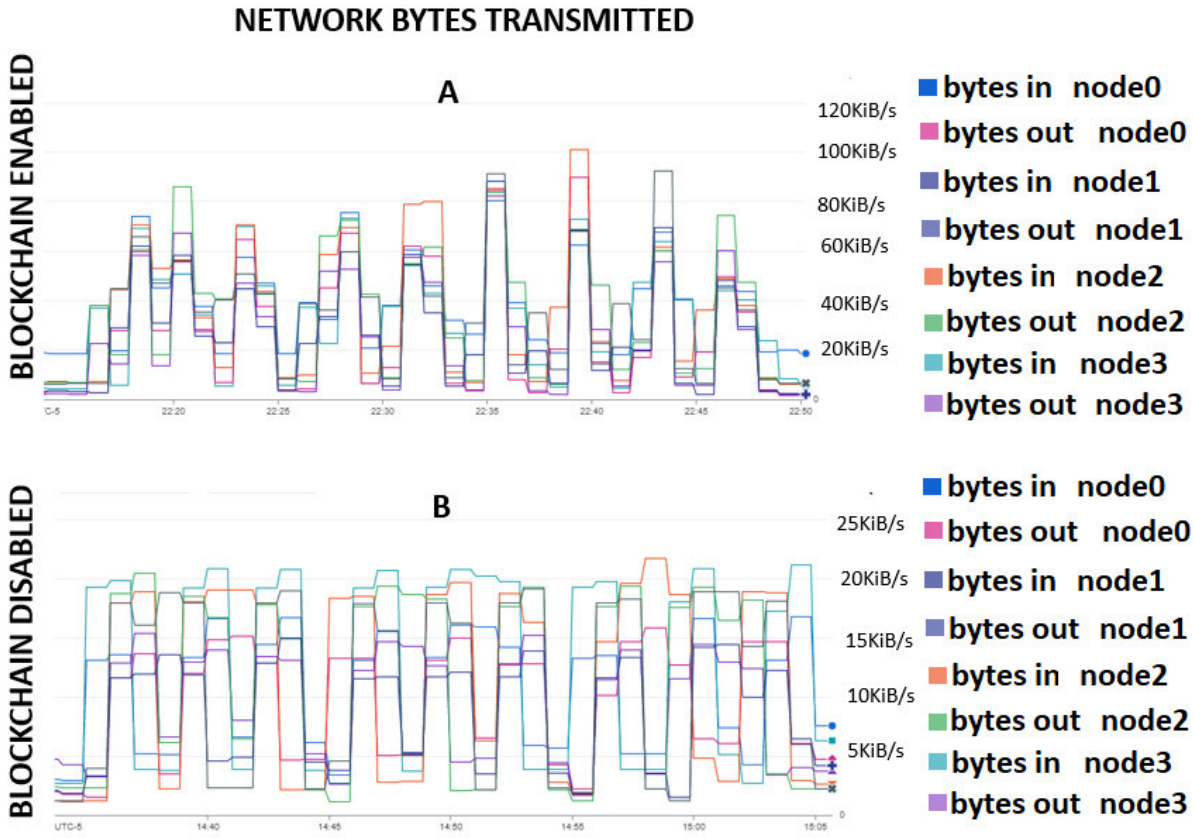


Figura 17 - Bytes de red transmitidos con 1000 transacciones

La Figura 18 - A muestra las operaciones de Lectura / Escritura en disco luego de ejecutar 3 conjuntos de 1000 inserciones, 3 conjuntos de 1000 actualizaciones y 3 conjuntos de 1000 eliminaciones con la blockchain activada y la Figura 18 - B muestra los bytes de red transmitidos para el mismo caso con la blockchain desactivada.

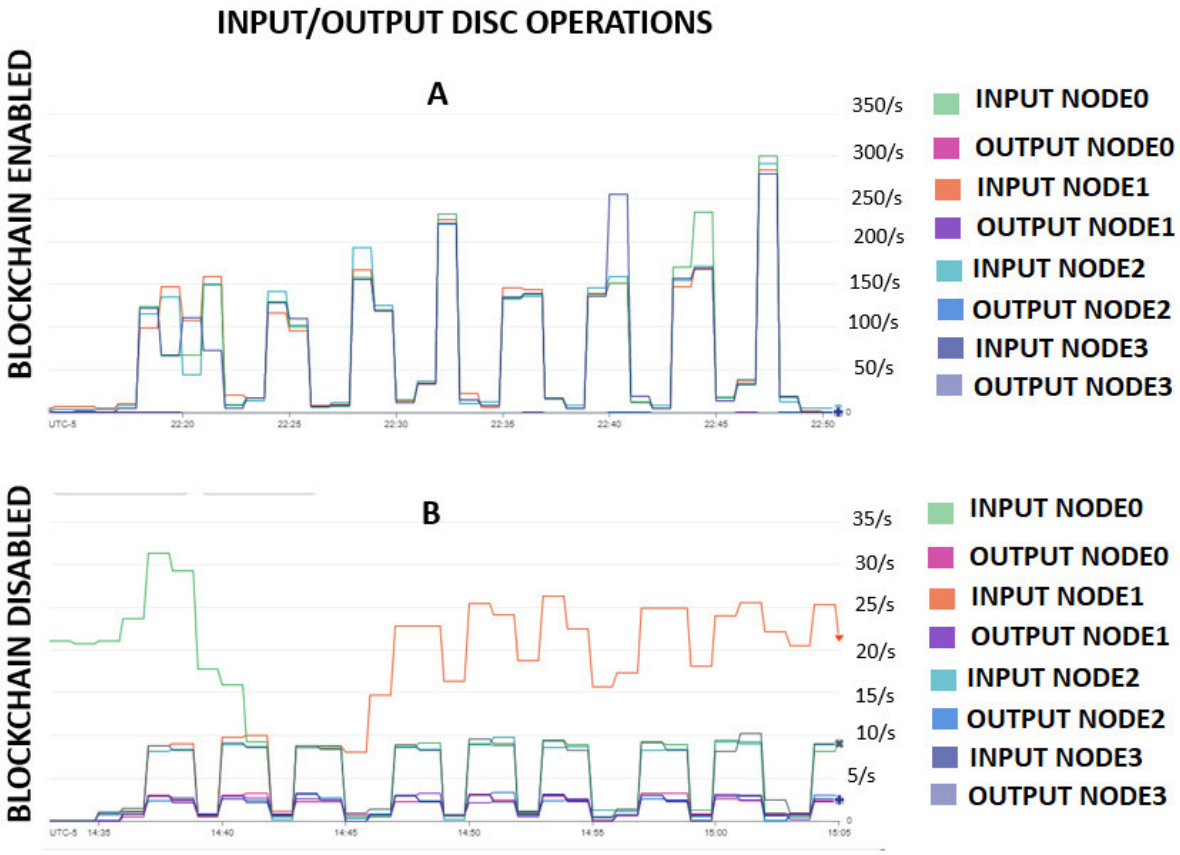


Figura 18 - Operaciones de Lectura / Escritura en disco con 1000 transacciones

Al ejecutar 1000 transacciones, el uso de procesador y los bytes transmitidos en la red se multiplican aproximadamente por 5, mientras que las operaciones de lectura / escritura de disco se multiplican aproximadamente por 10 con la blockchain activa.

En cuanto al rendimiento de la arquitectura, la solución propuesta parece ser factible de ser utilizada en ambientes productivos siempre y cuando se cuente con servidores lo suficientemente potentes.

3.2.2. Consistencia de tiempo

La consistencia de tiempo se refiere a la verificación del orden de las transacciones, es decir, que la blockchain muestre las transacciones en el mismo orden en el que llegaron a la base de datos.

Para verificar esto, se desarrolló una aplicación frontend. Esta aplicación permite visualizar los datos registrados en la blockchain así como también los datos de la operación DML realizada sobre la base de datos y datos de auditoría adicionales como: El tiempo en el que la transacción salió de la máquina cliente (Apache JMeter), el tiempo

en el que la transacción llegó a la base de datos y, el tiempo en el que la transacción se registró en la blockchain.

En el Anexo IV se puede observar una captura de la aplicación desarrollada y en la Tabla 6 se presenta una muestra de los tiempos recolectados durante las pruebas.

Tabla 6 - Muestra de los tiempos recolectados

Transacción	**Servidor**	**Tiempo de aplicación**	**Tiempo de base de datos**	**Tiempo de blockchain**
4	sqlserver3	2021-05-25 19:05:27.829	2021-05-25 19:05:27.920	2021-05-25 19:05:27.934
3	sqlserver0	2021-05-25 19:05:27.829	2021-05-25 19:05:27.913	2021-05-25 19:05:27.925
2	sqlserver1	2021-05-25 19:05:27.704	2021-05-25 19:05:27.793	2021-05-25 19:05:27.813
1	sqlserver0	2021-05-25 19:05:27.704	2021-05-25 19:05:27.783	2021-05-25 19:05:27.800

En la Tabla 6, la columna *Tiempo de aplicación* indica el momento cuando la transacción se genera en el cliente (Apache JMeter), la columna *Tiempo de base de datos* indica el momento cuando la transacción es vista por la base de datos y la columna *Tiempo de blockchain* indica el momento cuando la transacción pasa la interfaz de comunicación con la blockchain. En esta tabla se observan dos ejemplos de transacciones concurrentes en diferentes servidores, las transacciones 1 y 2 y las transacciones 3 y 4. En estos ejemplos, el tiempo en el que las transacciones son generadas por el cliente es el mismo, sin embargo, llegan a la base de datos en diferentes momentos y, como se puede observar analizando la columna *Tiempo de blockchain*, el orden en el que las transacciones son vistas por la base de datos se conserva en la blockchain, que es precisamente lo que se busca con la implementación de la arquitectura propuesta.

La comunicación entre los diferentes componentes involucrados (aplicación cliente, base de datos y blockchain) se realiza en promedio en tiempos menores a las 100 milésimas de segundo, lo cual es un tiempo muy aceptable.

4. RESULTADOS Y DISCUSIÓN

En esta sección se realizará el análisis de la experimentación del funcionamiento de la arquitectura enfocándose principalmente en la integridad transaccional, en la viabilidad y en la implementación de la solución.

4.1. Resultados

De acuerdo con las pruebas realizadas, la arquitectura propuesta cumple con el objetivo de generar una línea de tiempo global de transacciones generadas en un ambiente distribuido. Si se ejecutan transacciones simultáneamente en varios servidores de base de datos, la arquitectura es capaz de generar en una blockchain un registro consolidado que incluya las transacciones de todos los servidores conservando el orden en el que estas arribaron a la base de datos.

En lo que se refiere al rendimiento de la solución, los resultados muestran que la arquitectura si puede llegar a ser aplicable en ambientes productivos siempre y cuando se cuente con servidores con los recursos suficientes ya que en pruebas realizadas sobre máquinas virtuales con 4 GB de memoria RAM y procesador compartido, se llegó a un estado de fallo irrecuperable de los servidores, es decir que, el rendimiento de la solución está altamente ligado a las características de los servidores.

En la evaluación se evidencia que la comunicación entre los componentes se realiza en tiempos lo bastante buenos como para considerar la implementación de la solución en ambientes de producción, además, se determina que no existe una diferencia representativa en el rendimiento entre inserciones, actualizaciones y eliminaciones.

4.2. Discusiones

Uno de los inconvenientes que podría presentar la implementación de la solución en ambientes productivos es la de no contar con servidores con los recursos suficientes a nivel de hardware como para suportar el funcionamiento continuo de la blockchain, sin embargo, si consideramos que actualmente toda empresa mediana o grande cuenta con servidores bastante potentes, este aspecto no llegaría a representar un problema para la implementación de la arquitectura. Incluso, si no se cuenta con servidores potentes, se podría optar por implementar la arquitectura en algún servicio en la nube como se lo hizo

para este proyecto. Las plataformas como Microsoft Azure, Google Cloud o Amazon Web Services ofrecen precios bastante asequibles en la actualidad.

Técnicamente, la implementación de la arquitectura no representa un esfuerzo demasiado grande gracias a que las tecnologías de blockchain existentes a la fecha proveen mecanismos de fácil implementación. Sin embargo, a nivel económico si pudiese representar un esfuerzo mayor si se desea contar con una solución de blockchain de pago, ya que a pesar de que en el presente trabajo se utilizó una solución libre y gratuita, existen proyectos de blockchain de pago que incluyen planes de soporte y asesoramiento.

El uso de la tecnología de blockchain puede ir más allá de las criptomonedas. Hay muchos proyectos de blockchain en curso que se pueden acoplar a casi cualquier necesidad empresarial; Además, la mayoría de estos proyectos pueden interactuar fácilmente con otras tecnologías, por lo general, mediante servicios REST. En este proyecto, se valida el uso de blockchain para generar datos de auditoría en una base de datos.

5. CONCLUSIONES Y RECOMENDACIONES

En esta sección se presentan las conclusiones resultantes de la elaboración de este proyecto y recomendaciones que permitirían mejorar el presente trabajo.

5.1. Conclusiones

Mediante este trabajo, se logró determinar que si es posible implementar una arquitectura basada en blockchain que permita secuenciar la ocurrencia de eventos en bases de datos para fines de auditoría.

Se logró determinar que mediante blockchain los requerimientos funcionales para auditorías de bases de datos si pueden ser cubiertos, específicamente en lo que se refiere a la secuencia correcta de transacciones en bases de datos distribuidas.

La evaluación de rendimiento realizada sobre la arquitectura indica que el uso de blockchain si pudiese estar justificado para implementaciones referentes a auditorías de base de datos.

El hardware puede llegar a ser un limitante para la implementación de proyectos como el presentado en este trabajo, ya que, el funcionamiento de una blockchain permanentemente en una red requerirá equipos bastante potentes.

Luego de esta investigación, queda claro que blockchain puede empezar a ser utilizado en otro tipo de proyectos que quisiesen involucrase en esta tecnología que, si bien no es tan reciente, su uso se ha visto limitado a criptomonedas y poco más. Además, a medida que evoluciona la tecnología blockchain, su implementación se hace cada vez más accesible.

5.2. Recomendaciones

La solución presentada en este trabajo debe ser mejorada en cuanto a su desempeño para permitir manejar un mayor número de transacciones sin la necesidad de mejorar sustancialmente las características de los servidores. Este aspecto fue uno de los inconvenientes encontrados durante las pruebas de la arquitectura.

Hay mucho trabajo en lo que se refiere a la adoptación de la tecnología blockchain en áreas diferentes a la de las criptomonedas como por ejemplo en el Internet de las cosas.

Se presenta un gran desafío para la tecnología de blockchain frente a la computación cuántica ya que la tecnología blockchain actual trabaja en base a algoritmos que usan funciones hash para mantener su seguridad y la computación cuántica podría violar la seguridad de estos algoritmos fácilmente. En necesario que blockchain evolucione junto a esta nueva tecnología emergente.

REFERENCIAS BIBLIOGRÁFICAS

[1] S. Mu, Y. Cui, Y. Zhang, W. Lloyd y J. Li, «Extracting more concurrency from distributed transactions,» de *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14*, 2014.

[2] P. Bernstein y N. Goodman, «Concurrency control in distributed database systems,» *ACM Computing Surveys (CSUR),* vol. 13, nº 2, pp. 185-221, 1981.

[3] X. Jiang, «The Application of Blockchain in Auditing,» de *2018 2nd International Conference on Education Science and Economic Management (ICESEM 2018)*, 2018.

[4] The Guardian, «Newest cyber threat will be data manipulation, US intelligence chief says,» 2015. [En línea]. Available: https://www.theguardian.com/technology/2015/sep/10/cyber-threat-data-manipulation-us-intelligence-chief. [Último acceso: 7 Julio 2020].

[5] Linux Foundation, «Hyperledger Sawtooth,» 2020. [En línea]. Available: https://sawtooth.hyperledger.org/. [Último acceso: 17 Enero 2021].

[6] The Apache Software Foundation, «Apache JMeter,» [En línea]. Available: https://jmeter.apache.org/.

[7] A. Hevner, S. March, J. Park y S. Ram, «Design science in information systems research,» *MIS quarterly,* pp. 75-105, 2004.

[8] C. Mullins, Database administration: the complete guide to practices and procedures, Addison-Wesley Professional, 2002.

[9] L. Yang, «Teaching database security and auditing,» de *Proceedings of the 40th ACM technical symposium on Computer science education*, 2009.

[10] A. M. a. W.-M. R. Costello, «The impact of financial reporting quality on debt contracting: Evidence from internal control weakness reports,» *Journal of Accounting Research,* vol. 49, nº 1, pp. 97-136, 2011.

[11] V. Michael, «The Requirement for a Proactive Audit Program,» 31 Enero 2019. [En línea]. Available: https://blog.volkovlaw.com/2019/01/the-requirement-for-a-proactive-audit-program/. [Último acceso: 31 Agosto 2020].

[12] Association of Chief Police Officers, «ACPO Good Practice Guide for Digital Evidence,» 2012. [En línea]. Available: https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf. [Último acceso: 22 Septiembre 2020].

[13] D. A. a. J. A. Flores, «Implementing chain of custody requirements in database audit records for forensic purposes,» de *2017 IEEE Trustcom/BigDataSE/ICESS*, 2017.

[14] R. Harding, D. Van Aken, A. Pavlo y M. Stonebraker, «An evaluation of distributed

concurrency control,» *Proceedings of the VLDB Endowment,* vol. 10, nº 5, pp. 553-564, 2017.

[15] BBVA, «¿Cuál es la diferencia entre una DLT y 'blockchain'?,» 2018. [En línea]. Available: https://www.bbva.com/es/diferencia-dlt-blockchain/. [Último acceso: 28 Junio 2020].

[16] S. Zhang y J.-H. Lee, «Analysis of the main consensus protocols of blockchain,» *ICT Express,* 2019.

[17] I. B. Damgård, «A design principle for hash functions,» de *Conference on the Theory and Application of Cryptology*, 1989.

[18] Binance Academy, «What Is a Blockchain Consensus Algorithm?,» Diciembre 2020. [En línea]. Available: https://academy.binance.com/blockchain/what-is-a-blockchain-consensus-algorithm. [Último acceso: 17 Junio 2020].

[19] S. a. B. A. Nakamoto, «A peer-to-peer electronic cash system,» *Bitcoin.--URL: https://bitcoin.org/bitcoin.pdf,* 2008.

[20] C. Hammerschmidt, «Consensus in Blockchain Systems. In Short,» 2017. [En línea]. Available: https://medium.com/@chrshmmmr/consensus-in-blockchain-systems-in-short-691fc7d1fefe. [Último acceso: 28 Junio 2020].

[21] J. Franke, «Blockchain Consensus Algorithms – Proof of Anything?,» 2017. [En línea]. Available: https://jornfranke.wordpress.com/2017/11/18/blockchain-consensus-algorithms-proof-of-anything/. [Último acceso: 28 Junio 2020].

[22] Z. Shijie y L. Jong-Hyouk, «Analysis of the main consensus protocols of blockchain,» *ScienceDirect,* 2019.

[23] Binance Academy, «Byzantine Fault Tolerance Explained,» Enero 2021. [En línea]. Available: https://academy.binance.com/en/articles/byzantine-fault-tolerance-explained.

[24] M. Castro y B. Liskov, «Practical byzantine fault tolerance,» de *Symposium on Operating Systems Design and Implementation*, 1999.

[25] Intel Corporation, «Sawtooth Enterprise Blockchain,» [En línea]. Available: https://sawtooth.hyperledger.org/examples/. [Último acceso: 21 Junio 2021].

[26] K. Olson, M. Bowman, J. Mitchell, S. Amundson, D. Middleton y C. Montgomery, «Sawtooth: An Introduction,» *The Linux Foundation,* 2018.

[27] R. Kravchenko, «Hyperledger Sawtooth as a development framework: architecture overview,» 2018. [En línea]. Available: https://medium.com/remme/hyperledger-sawtooth-as-a-development-framework-architecture-overview-4947ec81fa50. [Último acceso: 20 Diciembre 2020].

[28] R. Marvin, «Blockchain: The Invisible Technology That's Changing the World,» 30 Agosto 2017. [En línea]. Available: https://au.pcmag.com/features/46389/blockchain-the-invisible-technology-thats-changing-the-world. [Último acceso: 20 Diciembre 2020].

[29] S. M. a. M. U. S. Groomer, «Continuous Auditing of Database Applications: An Embedded

Audit Module Approach1,» *Continuous auditing,* 2018.

[30] I. Traiger, J. Gray, C. Galtieri y B. Lindsay, «Transactions and consistency in distributed database systems,» *ACM Transactions on Database Systems (TODS),* vol. 7, nº 3, pp. 323-342, 1982.

[31] B. Ampel, M. Patton y H. Chen, «Performance modeling of hyperledger sawtooth blockchain,» *2019 IEEE International Conference on Intelligence and Security Informatics (ISI),* pp. 59-61, 2019.

[32] V. Mansur y R. Sujatha, «Hyperledger Sawtooth Blockchain-Iot E-Provenance Platform for Pharmaceuticals,» *International Journal of Engineering and Advanced Technology,* 2018.

[33] A. Belloum, «Using blockchain to validate audit trail data in private business applications,» *University of Amsterdam,* 2018.

[34] V. Buterin y Otros, «A next-generation smart contract and decentralized application platform,» *white paper,* 2014.

[35] A. M. Rozario y C. Thomas, «Reengineering the audit with blockchain and smart contracts,» *Journal of Emerging Technologies in Accounting,* pp. 21-35, 2019.

[36] KPMG, «KPMG Technology Industry Innovation Survey: Blockchain,» 2019. [En línea]. Available: https://assets.kpmg/content/dam/kpmg/us/pdf/2019/02/blockchain-tech-survey-2019-infographic.pdf. [Último acceso: 28 Junio 2020].

[37] The Linux Foundation, «Hyperledger,» [En línea]. Available: https://wiki.hyperledger.org/. [Último acceso: 7 Septiembre 2021].

[38] bigchaindb, [En línea]. Available: https://www.bigchaindb.com/. [Último acceso: 7 Septiembre 2021].

[39] Google, «Google Cloud Platform Compute Engine,» [En línea]. Available: https://cloud.google.com/compute.

[40] A. Arshad, «SQL Server Service Broker - An Introduction,» 6 Mayo 2010. [En línea]. Available: https://tinyurl.com/48tyhvm4.

[41] Microsoft, «SQL Server Service Broker,» 17 Marzo 2021. [En línea]. Available: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-service-broker.

[42] Grafana, «Grafana: The open observability platform,» [En línea]. Available: https://grafana.com.

ANEXOS

Anexo I – Guía técnica de implementación de la solución

Para la implementación de la solución que se propone, se usaron servidores en la plataforma Google Cloud Platform, esto no es un requisito, la solución puede ser implementada en máquinas virtuales o servidores físicos siempre y cuando se puedan conectar mediante red.

Configuración de los servidores en Google Cloud Platform

Para el caso particular de servidores en Google Cloud Platform, es necesario tener una cuenta de Google y tener un proyecto creado en https://console.cloud.google.com/.

En esta guía se obviará el proceso de creación de los servidores y configuraciones de IP's ya que éste podría variar con el tiempo.

El código desarrollado se aplica a una Blockchain de 4 servidores, pero es fácilmente adaptable para cualquier número de servidores, siempre y cuando sean 4 o más, ya que el protocolo de consenso usado requiere mínimo 4 nodos. Se deberían tener al menos 4 nodos para la blockchain (en este caso con Ubuntu 20.04, se podría usar cualquier sistema operativo en el que se pueda instalar el gestor de contenedores Docker y SQL Server 2005 o superior) y 1 nodo para que cumpla las funciones de cliente (Windows Server 2019).

Nombre ↑	Zona	Tipo de máquina	IP interna	Conectar
client	us-east1-b	e2-medium	10.142.0.9 (nic0)	RDP ▾
node0	us-east1-b	e2-standard-4	10.142.0.2 (nic0)	SSH ▾
node1	us-east1-b	e2-standard-4	10.142.0.3 (nic0)	SSH ▾
node2	us-east1-b	e2-standard-4	10.142.0.4 (nic0)	SSH ▾
node3	us-east1-b	e2-standard-4	10.142.0.5 (nic0)	SSH ▾

Cada una de las máquinas tiene habilitados los siguientes puertos en el firewall para permitir la comunicación.

Puerto	**Aplicación**
1433	SQL Server
80	Servidor nginx
4000	Api del cliente de blockchain
3000	Grafana
8086	Influxdb

22	SSH

Conectados al **nodo 0** instalamos git y clonamos el repositorio de la solución con los siguientes comandos:

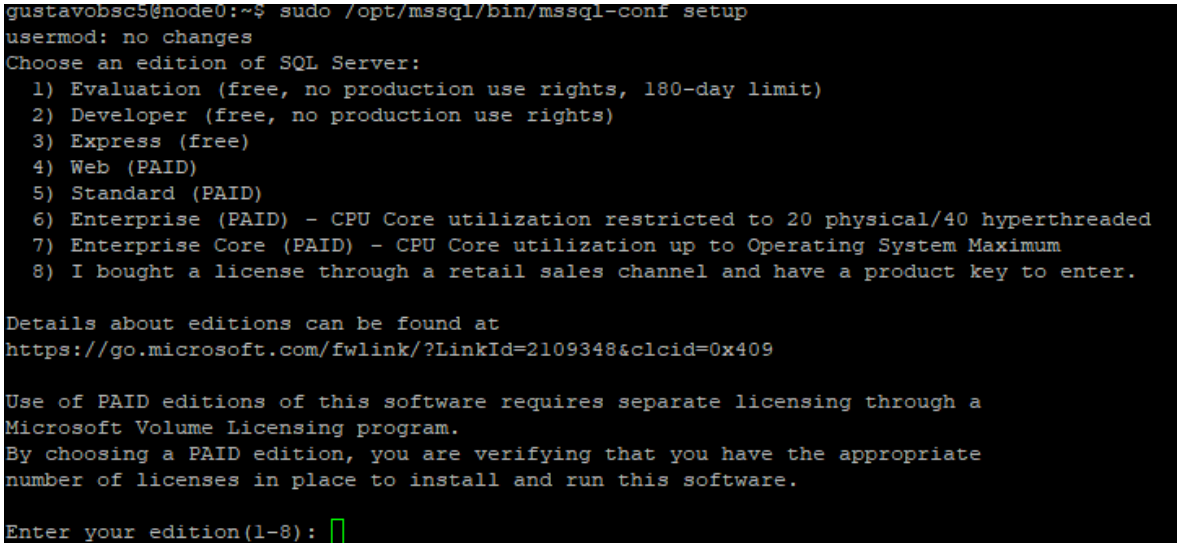
- sudo apt-get update
- sudo apt-get install git
- cd ~
- git clone https://github.com/gustavoalcivar/blockchain-based_architecture_to_sequence_distributed_database_events_occurrence.git

El último comando descargará el código fuente y las configuraciones de la solución. Dentro de la carteta descargada, se deben de configurar las IP's de los servidores en el directorio "ips".

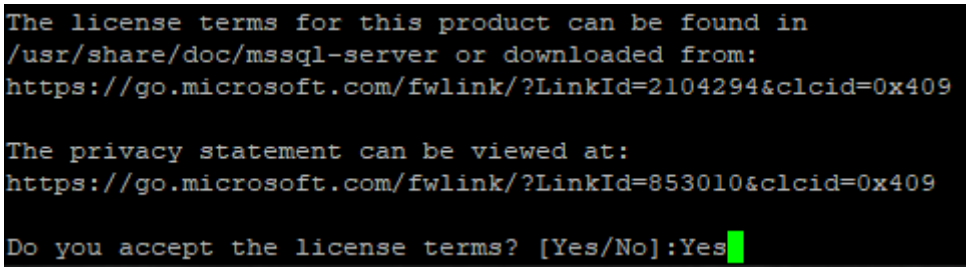
Instalación de SQL Server

Esta instalación se la debe realizar en **todos los nodos** de la blockchain. Para la instalación de SQL Server, seguir las instrucciones de https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver15 o ejecutar los siguientes comandos (para Ubuntu 20.04).

- wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add –
- sudo add-apt-repository "$(wget -qO-https://packages.microsoft.com/config/ubuntu/20.04/mssql-server-2019.list)"
- sudo apt-get update
- sudo apt-get install -y mssql-server
- sudo /opt/mssql/bin/mssql-conf setup



Para esta prueba se usó la versión de evaluación (1). Luego, se debe aceptar la licencia.



Configuramos la contraseña para el usuario “sa” y de esta manera queda instalado el gestor de base de datos en cada uno de los servidores.

Configuración de los nodos para trabajar de forma distribuida

Para configurar los nodos para trabajar de forma distribuida es necesario tener acceso a los servidores de base de datos mediante Microsoft SQL Server Management Studio o cualquier otro gestor que permita ejecutar consultas en las bases de datos y acceso a la línea de comandos de cada uno de los servidores.

Lo primero es configurar **Microsoft Distributed Transaction Coordinator**, mediante la guía publicada en https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-configure-msdtc?view=sql-server-ver15. Luego, se debe desplegar Microsoft Distributed Transaction Coordinator para transacciones distribuidas en “SQL Server Always On Availability Groups”, siguiendo la guía publicada en https://www.sqlshack.com/deploy-msdtc-for-distributed-transactions-in-sql-server-always-on-availability-groups/.

Generación de la base de datos de prueba

Dentro del directorio de la solución, se provee un script para generar una base de datos de prueba. Este script se encuentra en **./generate_db_script/database.sql**.

```
CREATE DATABASE mybank
go

use mybank
go

CREATE TABLE clientes (id INT IDENTITY(1,1) PRIMARY KEY, nombre VARCHAR(50), apellido VARCHAR(50), telefono VARCHAR(20), correo VARCHAR(50))
CREATE TABLE tipo_transaccion (id INT IDENTITY(1,1) PRIMARY KEY, descripcion VARCHAR(255))
CREATE TABLE cuentas_bancarias (id INT IDENTITY(1,1) PRIMARY KEY, moneda VARCHAR(10), id_cliente INT, saldo FLOAT, FOREIGN KEY (id_cliente)
CREATE TABLE transacciones (id INT IDENTITY(1,1) PRIMARY KEY, id_cuenta_bancaria INT, id_tipo_transaccion INT, monto FLOAT, FOREIGN KEY (id_
go

INSERT INTO clientes(nombre, apellido, telefono, correo) values ('Gustavo', 'Alcívar', '0960590338', 'gustavo.alcivar@epn.edu.ec')
go
INSERT INTO clientes(nombre, apellido, telefono, correo) values ('Alfonso', 'Rodríguez', '0960590339', 'gustavoalfonso05@hotmail.com')
go

INSERT INTO tipo_transaccion(descripcion) values('Depósito')
go
INSERT INTO tipo_transaccion(descripcion) values('Retiro')
go
```

```
%
Messages

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2021-08-05T16:45:08.0476849-05:00
```

A continuación, en los demás nodos, ejecutar únicamente las instrucciones para crear la base de datos.

```
CREATE DATABASE mybank
go
```

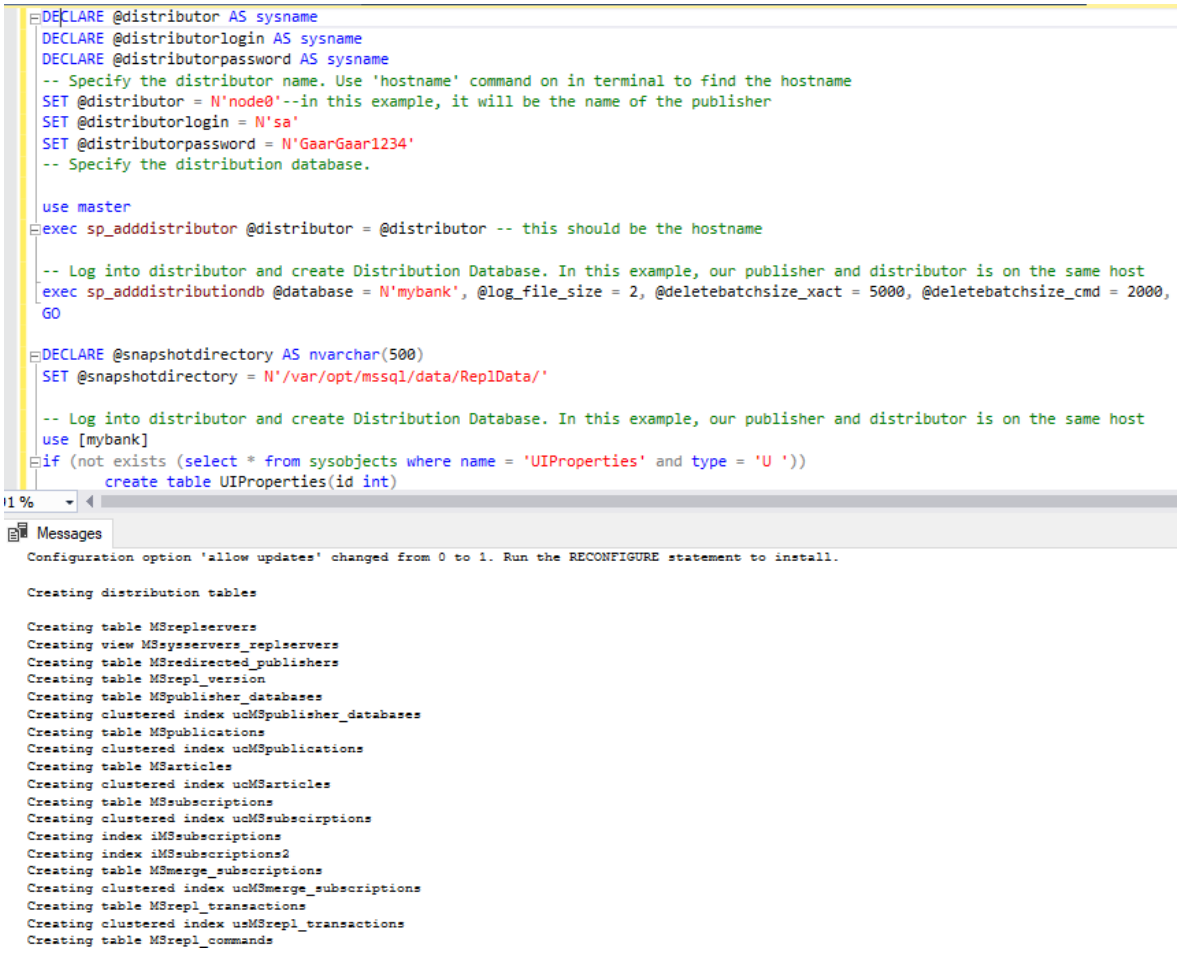
```
%
Messages

Commands completed successfully.

Completion time: 2021-08-05T16:46:51.1189467-05:00
```

Luego, en el **primer nodo**, crear una carpeta para las imágenes de la base de datos para que los demás nodos puedan leer y escribir en la base de datos, ejecutando los siguientes comandos.

- sudo mkdir /var/opt/mssql/data/ReplData/

- sudo chown mssql /var/opt/mssql/data/ReplData/
- sudo chgrp mssql /var/opt/mssql/data/ReplData/

Ahora, en el **primer nodo**, ejecutar el script **./replication_scripts/replication_script_1.sql** y luego el script **./replication_scripts/replication_script_2.sql**.



```
DECLARE @distributor AS sysname
DECLARE @distributorlogin AS sysname
DECLARE @distributorpassword AS sysname
-- Specify the distributor name. Use 'hostname' command on in terminal to find the hostname
SET @distributor = N'node0'--in this example, it will be the name of the publisher
SET @distributorlogin = N'sa'
SET @distributorpassword = N'GaarGaar1234'
-- Specify the distribution database.

use master
exec sp_adddistributor @distributor = @distributor -- this should be the hostname

-- Log into distributor and create Distribution Database. In this example, our publisher and distributor is on the same host
exec sp_adddistributiondb @database = N'mybank', @log_file_size = 2, @deletebatchsize_xact = 5000, @deletebatchsize_cmd = 2000,
GO

DECLARE @snapshotdirectory AS nvarchar(500)
SET @snapshotdirectory = N'/var/opt/mssql/data/ReplData/'

-- Log into distributor and create Distribution Database. In this example, our publisher and distributor is on the same host
use [mybank]
if (not exists (select * from sysobjects where name = 'UIProperties' and type = 'U '))
        create table UIProperties(id int)
```

```
Messages
Configuration option 'allow updates' changed from 0 to 1. Run the RECONFIGURE statement to install.

Creating distribution tables

Creating table MSreplservers
Creating view MSsysservers_replservers
Creating table MSredirected_publishers
Creating table MSrepl_version
Creating table MSpublisher_databases
Creating clustered index ucMSpublisher_databases
Creating table MSpublications
Creating clustered index ucMSpublications
Creating table MSarticles
Creating clustered index ucMSarticles
Creating table MSsubscriptions
Creating clustered index ucMSsubscirptions
Creating index iMSsubscriptions
Creating index iMSsubscriptions2
Creating table MSmerge_subscriptions
Creating clustered index ucMSmerge_subscriptions
Creating table MSrepl_transactions
Creating clustered index usMSrepl_transactions
Creating table MSrepl_commands
```

```
DECLARE @publisher AS sysname
DECLARE @distributorlogin AS sysname
DECLARE @distributorpassword AS sysname
-- Specify the distributor name. Use 'hostname' command on in terminal to find the hostname
SET @publisher = N'node0'
SET @distributorlogin = N'sa'
SET @distributorpassword = N'GaarGaar1234'
-- Specify the distribution database.

-- Adding the distribution publishers
exec sp_adddistpublisher @publisher = @publisher,
@distribution_db = N'mybank',
@security_mode = 0,
@login = @distributorlogin,
@password = @distributorpassword,
@working_directory = N'/var/opt/mssql/data/ReplData',
@trusted = N'false',
@thirdparty_flag = 0,
@publisher_type = N'MSSQLSERVER'
GO
```

%

Messages

```
Commands completed successfully.

Completion time: 2021-08-05T17:46:18.2555350-05:00
```

```
exec sp_addpublication
@publication = N'SnapshotRepl',
@description = N'Snapshot publication of database ''mybank'' from Publisher ''node0''.',
@retention = 0,
@allow_push = N'true',
@repl_freq = N'snapshot',
@status = N'active',
@independent_agent = N'true'

exec sp_addpublication_snapshot @publication = N'SnapshotRepl',
@frequency_type = 1,
@frequency_interval = 1,
@frequency_relative_interval = 1,
@frequency_recurrence_factor = 0,
@frequency_subday = 8,
@frequency_subday_interval = 1,
@active_start_time_of_day = 0,
@active_end_time_of_day = 235959,
@active_start_date = 0,
@active_end_date = 0,
@publisher_security_mode = 0,
@publisher_login = @publisherlogin,
@publisher_password = @publisherpassword
```

) %

Messages

```
Commands completed successfully.

Completion time: 2021-08-05T20:48:28.2553799-05:00
```

```
use [mybank]
exec sp_addarticle
@publication = N'SnapshotRepl',
@article = N'transacciones',
@source_owner = N'dbo',
@source_object = N'transacciones',
@type = N'logbased',
@description = null,
@creation_script = null,
@pre_creation_cmd = N'drop',
@schema_option = 0x000000000803509D,
@identityrangemanagementoption = N'manual',
@destination_table = N'transacciones',
@destination_owner = N'dbo',
@vertical_partition = N'false'
go

exec sp_addarticle
@publication = N'SnapshotRepl',
@article = N'tipo_transaccion',
@source_owner = N'dbo',
@source_object = N'tipo_transaccion',
@type = N'logbased',
```

```
%
Messages
Commands completed successfully.

Completion time: 2021-08-05T20:49:27.9903272-05:00
```

```
DECLARE @subscriber AS sysname
DECLARE @subscriber_db AS sysname
DECLARE @subscriberLogin AS sysname
DECLARE @subscriberPassword AS sysname
SET @subscriber = N'node1' -- for example, MSSQLSERVER
SET @subscriber_db = N'mybank'
SET @subscriberLogin = N'sa'
SET @subscriberPassword = N'GaarGaar1234'

use [mybank]
exec sp_addsubscription
@publication = N'SnapshotRepl',
@subscriber = @subscriber,
@destination_db = @subscriber_db,
@subscription_type = N'Push',
@sync_type = N'automatic',
@article = N'all',
@update_mode = N'read only',
@subscriber_type = 0

exec sp_addpushsubscription_agent
@publication = N'SnapshotRepl',
@subscriber = @subscriber,
@subscriber_db = @subscriber_db,
@subscriber_security_mode = 0,
@subscriber_login = @subscriberLogin,
@subscriber_password = @subscriberPassword,
@frequency_type = 1,
@frequency_interval = 0,
@frequency_relative_interval = 0,
```

Messages

```
Job 'node0-mybank-SnapshotRepl-NODE2-2' started successfully.
Warning: The distribution agent job has been implicitly created and will run under the SQL Server Agent Service Account.
Job 'node0-mybank-SnapshotRepl-NODE3-3' started successfully.
Warning: The distribution agent job has been implicitly created and will run under the SQL Server Agent Service Account.

Completion time: 2021-08-05T20:52:24.0992316-05:00
```

Generación de los objetos de la base de datos para la comunicación con la blockchain

Para generar el script de generación de objetos de la base de datos, es necesario modificar el fichero **./generate_db_script/config.js** de acuerdo a la base de datos y en una máquina con node js instalado, ejecutar **node generate_script.js** dentro del directorio **./generate_db_script/**. Esto generará un fichero **script_NOMBRE_BASE_DE_DATOS.sql** con el código SQL necesario para generar los objetos para la base de datos particular.

```
use mybank
go
ALTER DATABASE [mybank] SET ENABLE_BROKER WITH ROLLBACK IMMEDIATE;

-----------------------------------------------------------------------
BEGIN TRY
DROP EVENT NOTIFICATION EventNotificationTargetQueue
  ON QUEUE TargetQueue

DROP SERVICE ExternalActivatorService
DROP QUEUE ExternalActivatorQueue
DROP SERVICE TargetService
DROP SERVICE InitiatorService
DROP QUEUE TargetQueue
DROP QUEUE InitiatorQueue
DROP CONTRACT [http://audit_blockchail/Contract]
DROP MESSAGE TYPE [http://audit_blockchail/ResponseMessage]
DROP MESSAGE TYPE [http://audit_blockchail/RequestMessage]
END TRY
BEGIN CATCH
print 'CREATING NEW OBJECTS'
END CATCH
-----------------------------------------------------------------------

CREATE MESSAGE TYPE [http://audit_blockchail/RequestMessage]
VALIDATION = WELL_FORMED_XML

CREATE MESSAGE TYPE [http://audit_blockchail/ResponseMessage]
VALIDATION = WELL_FORMED_XML
```

10 %

Messages

```
CREATING NEW OBJECTS

Completion time: 2021-07-22T20:37:27.1455886-05:00
```

Instalación del software necesario para la ejecución de la blockchain

Para el funcionamiento de la blockchain, en necesario instalar Docker y Docker-Compose **en todos los nodos** con los siguientes comandos:

- sudo apt-get remove docker docker-engine docker.io containerd runc
- sudo apt-get update
- sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

- echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- sudo apt-get update
- sudo apt-get install docker-ce docker-ce-cli containerd.io
- sudo usermod -aG docker $USER
- sudo curl -L "https://github.com/docker/compose/releases/download/1.28.6/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
- sudo chmod +x /usr/local/bin/docker-compose

Configuración necesaria para la ejecución de la blockchain

Para la ejecución de la blockchain, es necesario compartir las claves públicas de los servidores, para ello, en el **nodo 0** ejecutar lo siguiente:

- sudo apt-get update
- sudo apt-get install -y nfs-kernel-server

Añadir las siguientes líneas en el fichero /**etc**/**exports** (reenplazar el usuario y las IP's).

/home/usuario/blockchain 10.142.0.3(rw,no_subtree_check)
/home/usuario/blockchain 10.142.0.4(rw,no_subtree_check)
/home/usuario/blockchain 10.142.0.5(rw,no_subtree_check)

Y ejecutar:

- sudo exportfs -arv
- sudo systemctl enable nfs-kernel-server
- sudo systemctl start nfs-kernel-server

Renombrar el directorio de la solución con el nombre "blockchain" con el comando:

- mv /home/usuario/blockchain-based_architecture_to_sequence_distributed_database_events_occurrence/ /home/usuario/blockchain

En los **demás nodos** ejecutar:

- sudo apt-get update
- sudo apt-get install -y nfs-common

- mkdir -p /home/usuario/blockchain/

Y añadir la siguiente línea al fichero /etc/fstab (reemplazar el usuario y las IP's).

- 10.142.0.2:/home/usuario/blockchain /home/usuario/blockchain/ nfs rw 0 0

10.142.0.2 corresponde a la IP del servidor en el que se clonó el repositorio de git.

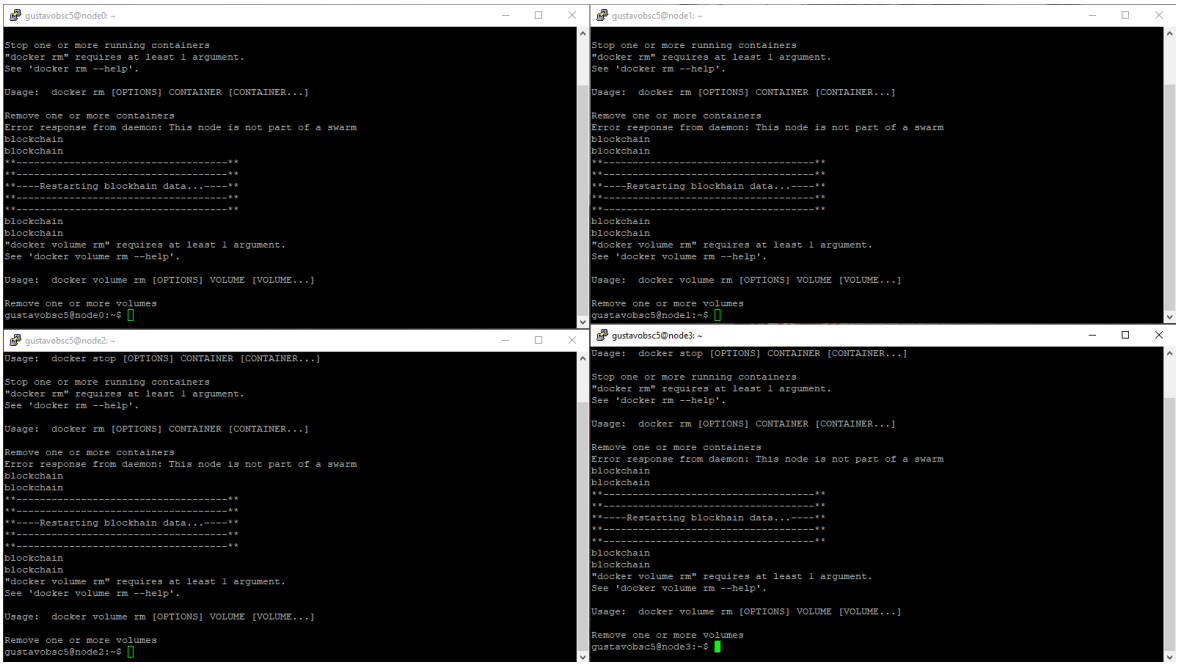
Luego, ejecutar:

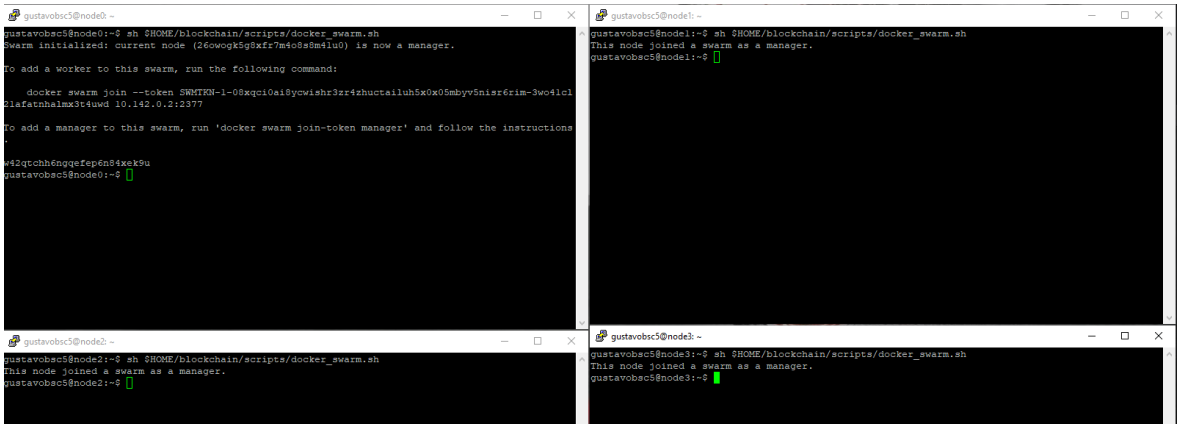
- sudo mount -a

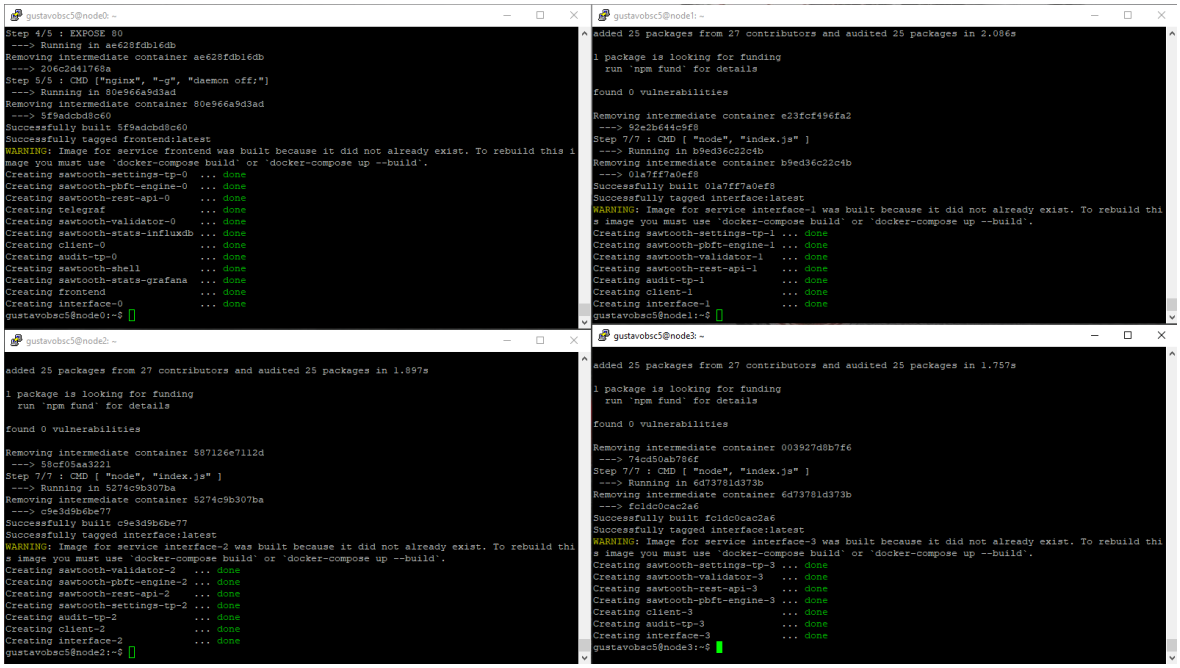
Ejecución de la blockchain

Es necesario volver a iniciar sesiones en los servidores para actualizar los permisos del usuario a Docker. Para iniciar la blockchain, ejecutar los siguientes comandos en cada uno de los nodos (un comando a la vez en cada servidor y esperando que cada comando finalice antes de ejecutar el siguiente).

- sh $HOME/blockchain/scripts/restart.sh
- sh $HOME/blockchain/scripts/docker_swarm.sh
- sh $HOME/blockchain/scripts/run.sh

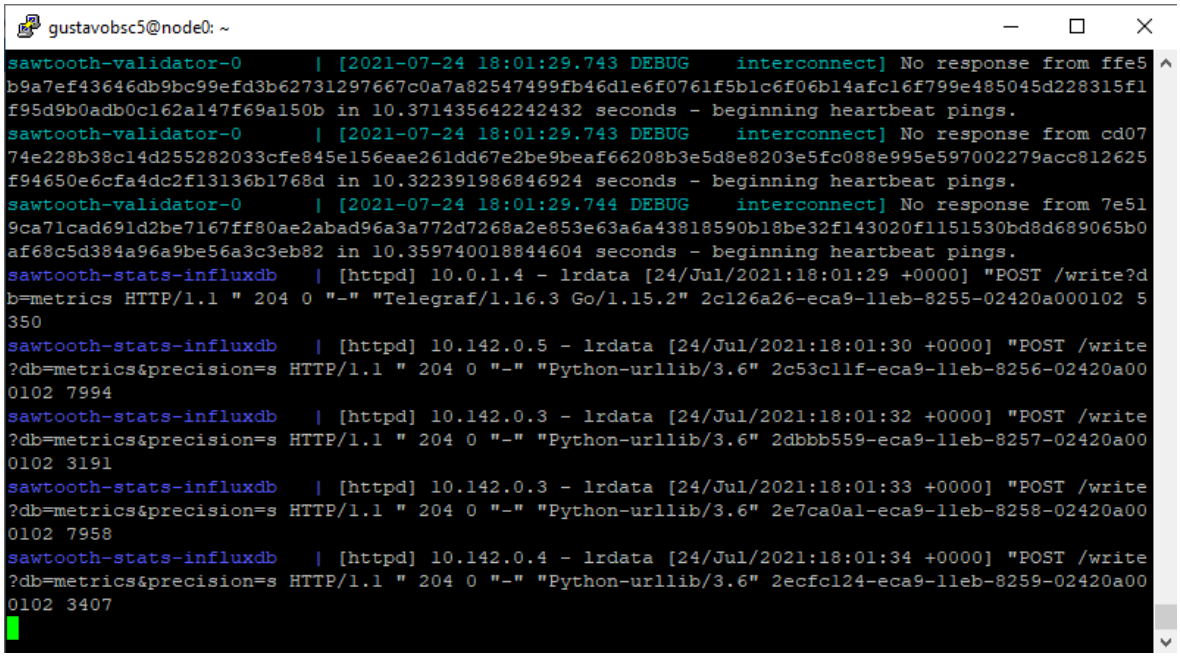






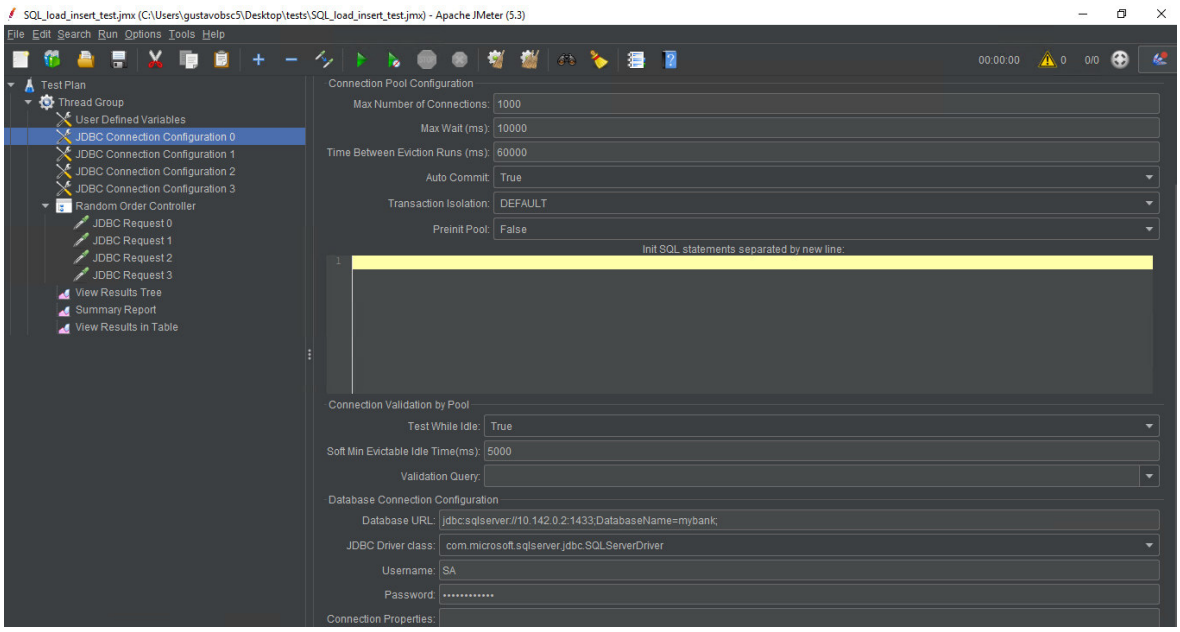
Para ver los logs de la blockchain, ejecutar el siguiente comando en cada uno de los servidores:

- sh $HOME/blockchain/scripts/logs.sh

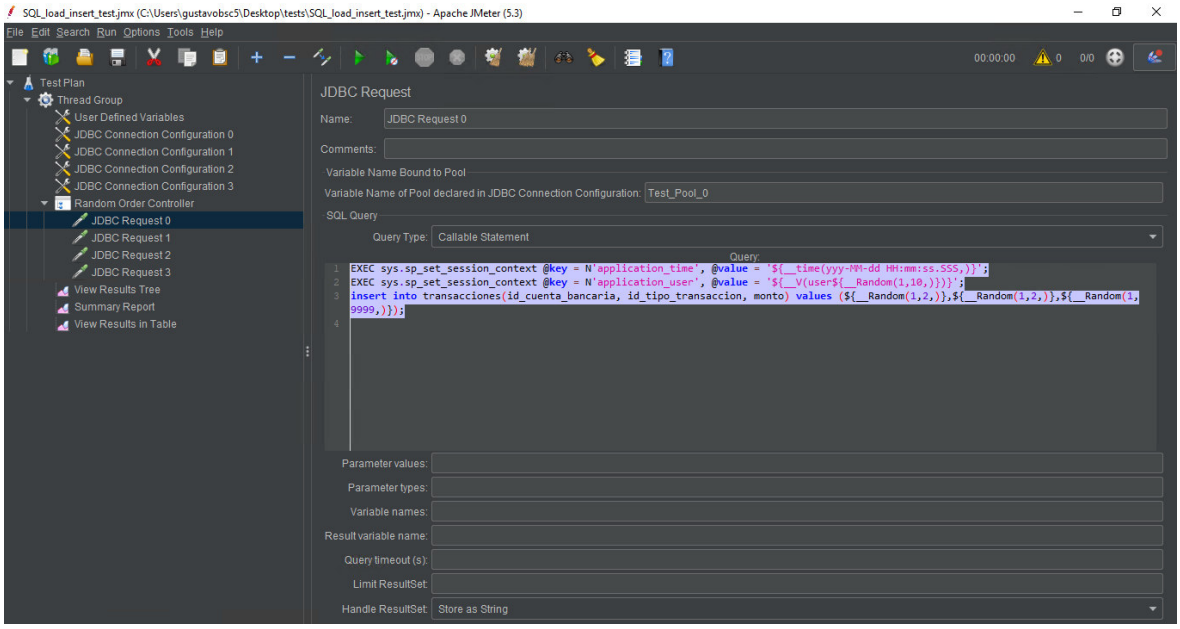


Configuración del cliente

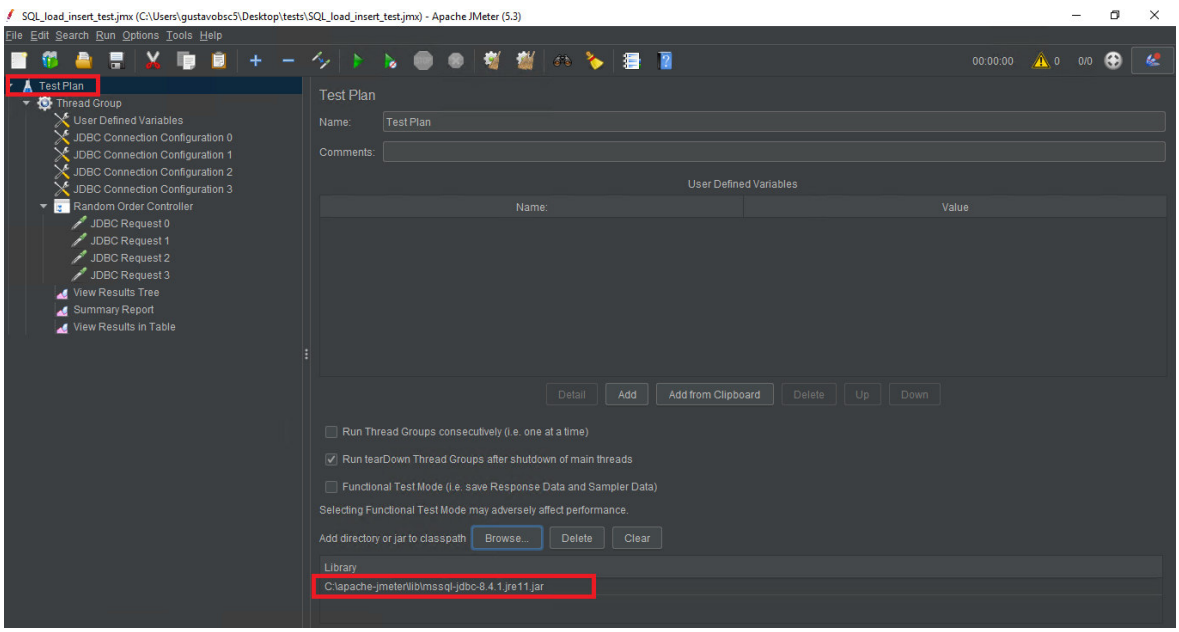
La máquina cliente es un Windows Server 2019 con el software JMeter y Java conectada a la misma red de los servidores de la blockchain. Los ficheros de las pruebas se encuentran en el repositorio de la solución, en el directorio **./jmeter/prod**. Es necesario copiar estos ficheros a la máquina cliente. Estas pruebas corresponden a inserciones, actualizaciones y eliminaciones.



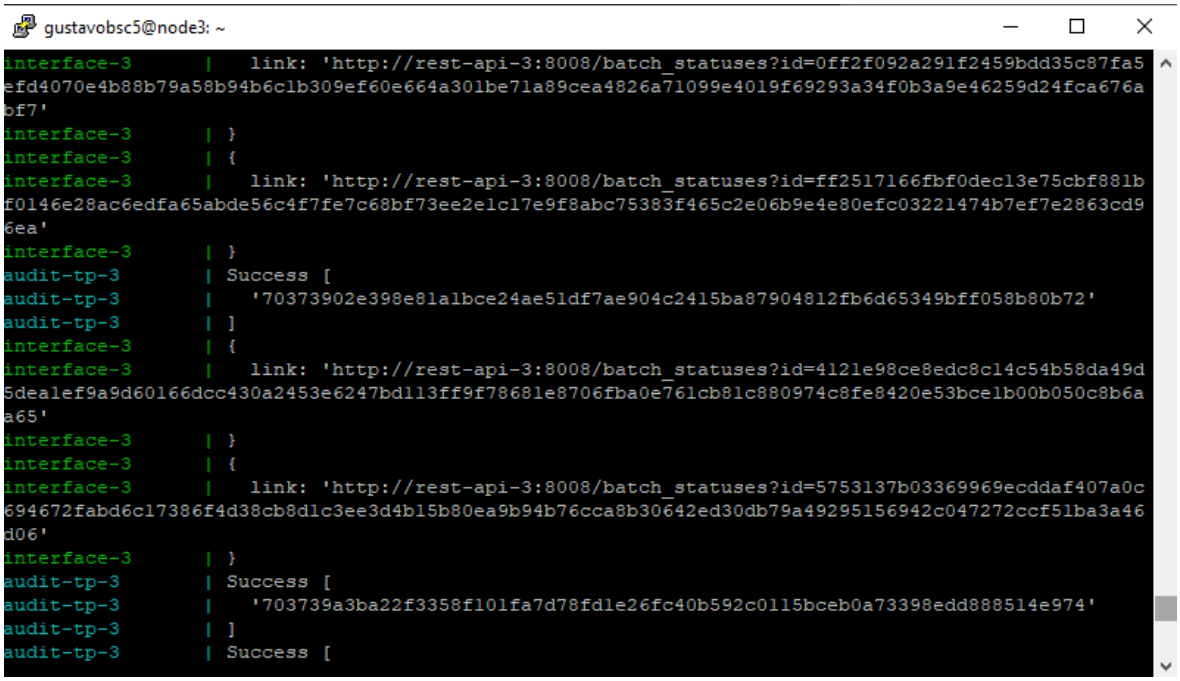
Cada uno de estos ficheros de pruebas, genera las variables de sesión del tiempo de ejecución, el usuario de aplicación y realizan la operación DML correspondiente en los 4 servidores aleatoriamente.



Para que la conexión con SQL Server funcione, se debe descargar el conector JDBC y configurarlo en el test plan de JMeter.

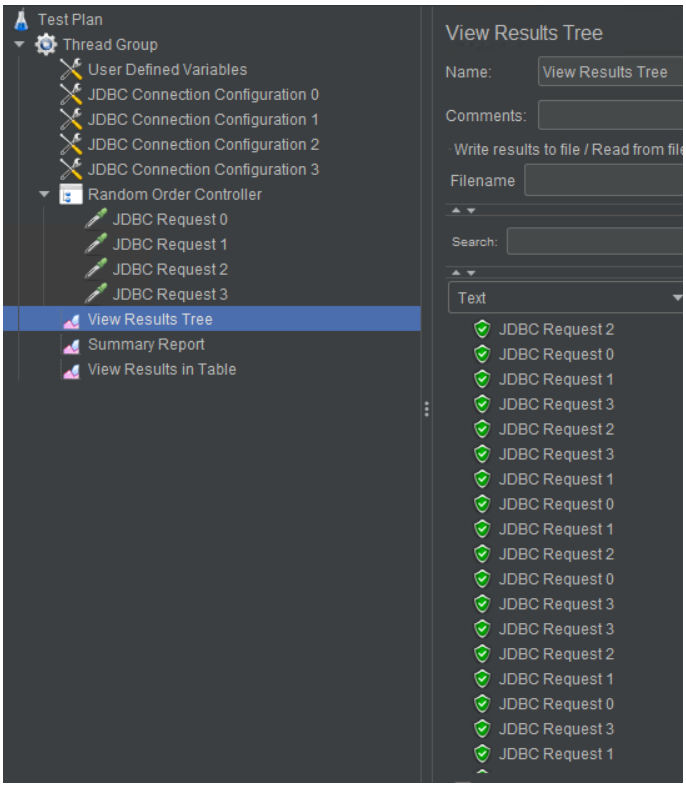


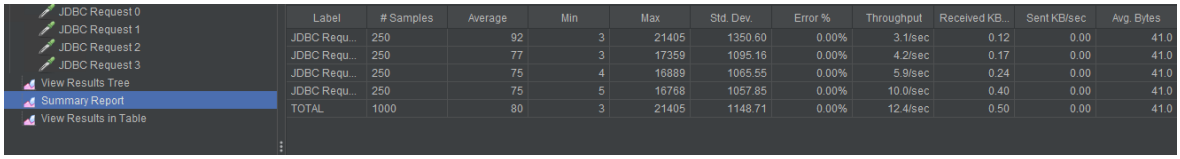
Finalmente, y asegurándose que la blockchain esté en ejecución, lanzamos las pruebas desde JMeter y, si se están mostrando los logs en los nodos de la blockchain, se observará algo como lo que se ve en la siguiente imagen.



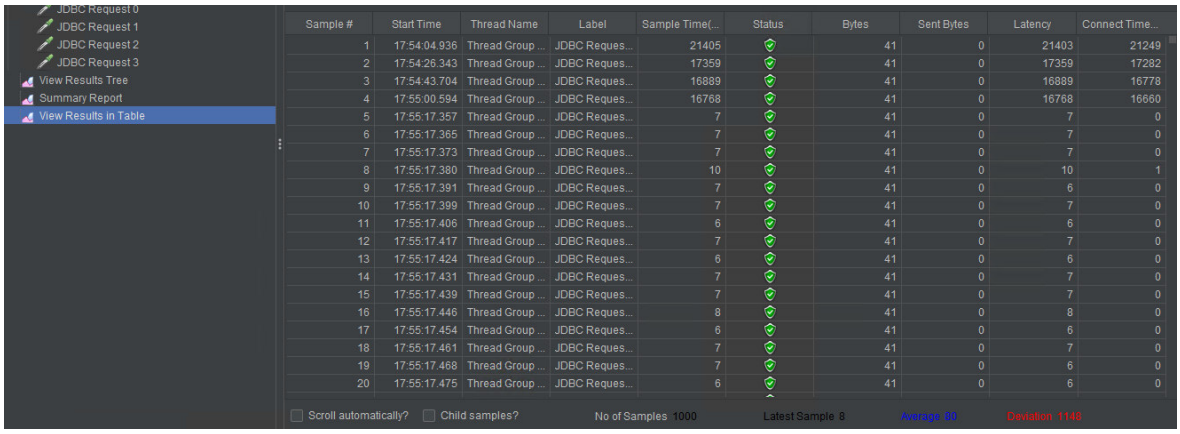
La imagen anterior indica que las transacciones fueron registradas correctamente en la blockchain.

Y en JMeter veremos los siguientes resultados:





Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
JDBC Requ...	250	92	3	21405	1350.60	0.00%	3.1/sec	0.12	0.00	41.0
JDBC Requ...	250	77	3	17359	1095.16	0.00%	4.2/sec	0.17	0.00	41.0
JDBC Requ...	250	75	4	16889	1065.55	0.00%	5.9/sec	0.24	0.00	41.0
JDBC Requ...	250	75	5	16768	1057.85	0.00%	10.0/sec	0.40	0.00	41.0
TOTAL	1000	80	3	21405	1148.71	0.00%	12.4/sec	0.50	0.00	41.0

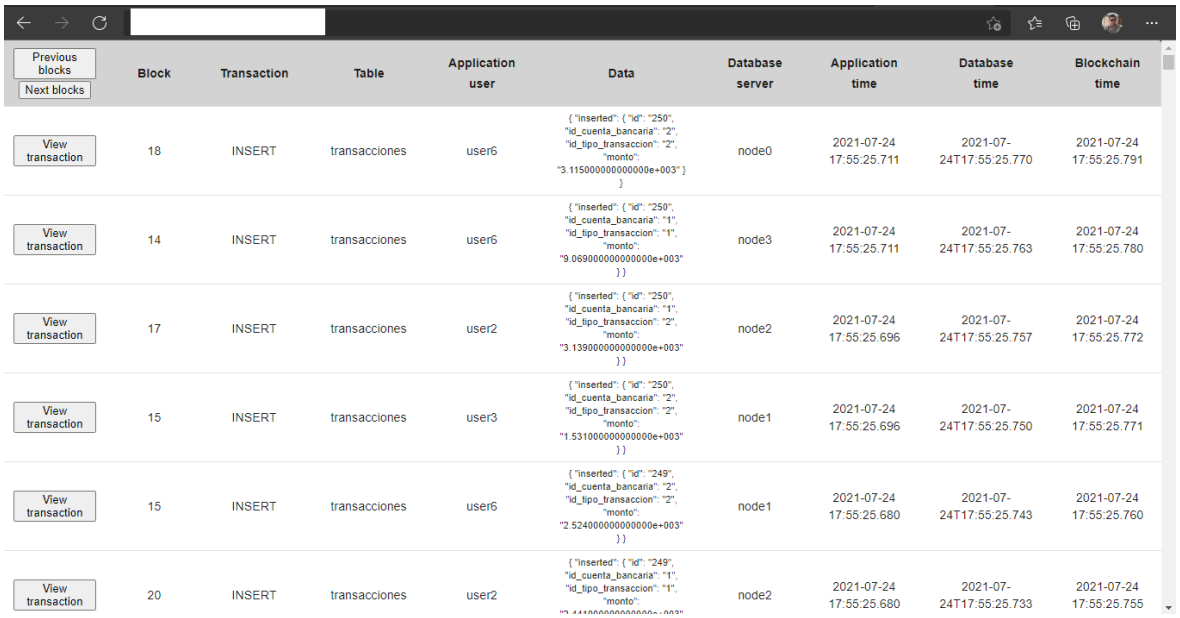


Sample #	Start Time	Thread Name	Label	Sample Time(...	Status	Bytes	Sent Bytes	Latency	Connect Time...
1	17:54:04.936	Thread Group ...	JDBC Reques...	21405		41	0	21403	21249
2	17:54:26.343	Thread Group ...	JDBC Reques...	17359		41	0	17359	17282
3	17:54:43.704	Thread Group ...	JDBC Reques...	16889		41	0	16889	16778
4	17:55:00.594	Thread Group ...	JDBC Reques...	16768		41	0	16768	16660
5	17:55:17.357	Thread Group ...	JDBC Reques...	7		41	0	7	0
6	17:55:17.365	Thread Group ...	JDBC Reques...	7		41	0	7	0
7	17:55:17.373	Thread Group ...	JDBC Reques...	7		41	0	7	0
8	17:55:17.380	Thread Group ...	JDBC Reques...	10		41	0	10	1
9	17:55:17.391	Thread Group ...	JDBC Reques...	7		41	0	6	0
10	17:55:17.399	Thread Group ...	JDBC Reques...	7		41	0	7	0
11	17:55:17.406	Thread Group ...	JDBC Reques...	6		41	0	6	0
12	17:55:17.417	Thread Group ...	JDBC Reques...	7		41	0	7	0
13	17:55:17.424	Thread Group ...	JDBC Reques...	6		41	0	6	0
14	17:55:17.431	Thread Group ...	JDBC Reques...	7		41	0	7	0
15	17:55:17.439	Thread Group ...	JDBC Reques...	7		41	0	7	0
16	17:55:17.446	Thread Group ...	JDBC Reques...	8		41	0	8	0
17	17:55:17.454	Thread Group ...	JDBC Reques...	6		41	0	6	0
18	17:55:17.461	Thread Group ...	JDBC Reques...	7		41	0	7	0
19	17:55:17.468	Thread Group ...	JDBC Reques...	7		41	0	6	0
20	17:55:17.475	Thread Group ...	JDBC Reques...	6		41	0	6	0

Scroll automatically? Child samples? No of Samples 1000 Latest Sample 8 Average 89 Deviation 1148

Visualización de datos registrados en la blockchain

Al ejecutar la blockchain, también se levanta un servidor nginx en un contenedor Docker con una aplicación web para poder ver las transacciones registradas, a esta se puede acceder visitando la ip pública del nodo 0 de la blockchain.



Previous blocks / Next blocks

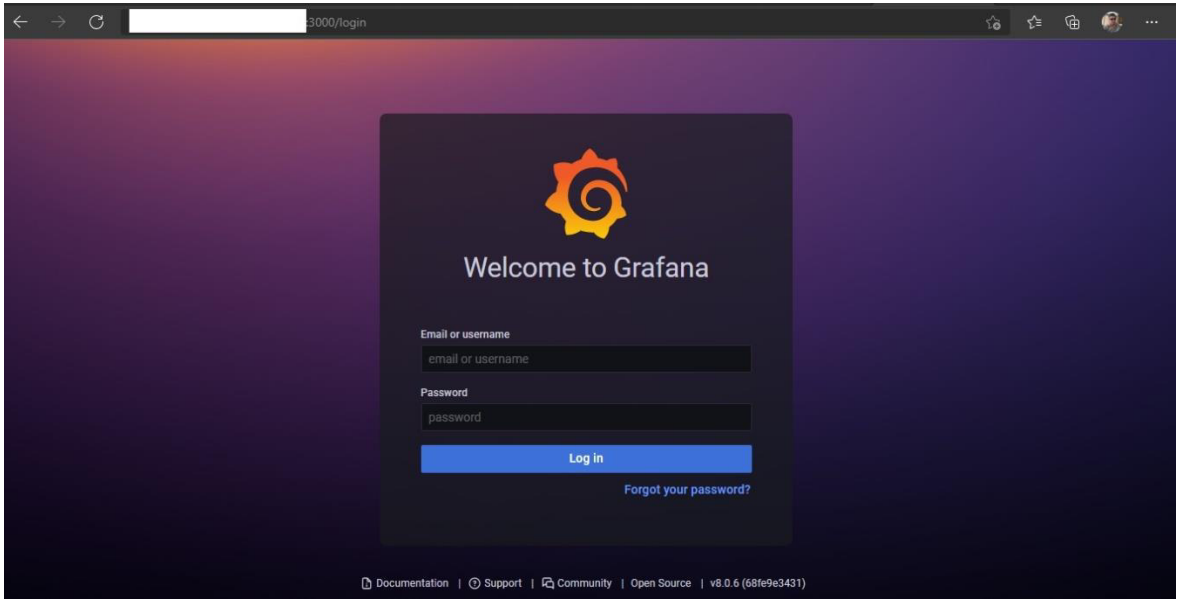
	Block	Transaction	Table	Application user	Data	Database server	Application time	Database time	Blockchain time
View transaction	18	INSERT	transacciones	user6	{ "inserted": { "id": "250", "id_cuenta_bancaria": "2", "id_tipo_transaccion": "2", "monto": "3.115000000000000e+003" } }	node0	2021-07-24 17:55:25.711	2021-07-24T17:55:25.770	2021-07-24 17:55:25.791
View transaction	14	INSERT	transacciones	user6	{ "inserted": { "id": "250", "id_cuenta_bancaria": "1", "id_tipo_transaccion": "1", "monto": "9.069000000000000e+003" } }	node3	2021-07-24 17:55:25.711	2021-07-24T17:55:25.763	2021-07-24 17:55:25.780
View transaction	17	INSERT	transacciones	user2	{ "inserted": { "id": "250", "id_cuenta_bancaria": "1", "id_tipo_transaccion": "2", "monto": "3.139000000000000e+003" } }	node2	2021-07-24 17:55:25.696	2021-07-24T17:55:25.757	2021-07-24 17:55:25.772
View transaction	15	INSERT	transacciones	user3	{ "inserted": { "id": "250", "id_cuenta_bancaria": "2", "id_tipo_transaccion": "2", "monto": "1.531000000000000e+003" } }	node1	2021-07-24 17:55:25.696	2021-07-24T17:55:25.750	2021-07-24 17:55:25.771
View transaction	15	INSERT	transacciones	user6	{ "inserted": { "id": "249", "id_cuenta_bancaria": "2", "id_tipo_transaccion": "2", "monto": "2.524000000000000e+003" } }	node1	2021-07-24 17:55:25.680	2021-07-24T17:55:25.743	2021-07-24 17:55:25.760
View transaction	20	INSERT	transacciones	user2	{ "inserted": { "id": "249", "id_cuenta_bancaria": "1", "id_tipo_transaccion": "1", "monto":	node2	2021-07-24 17:55:25.680	2021-07-24T17:55:25.733	2021-07-24 17:55:25.755

En esta aplicación se podrán observar todas las operaciones DML realizadas sobre la base de datos mientras haya estado en ejecución la blockchain. Este registro podría ser usado como evidencia de las operaciones realizadas en auditorías o casos de informática forense.

Anexo II – Pasos para configurar el tablero de Grafana

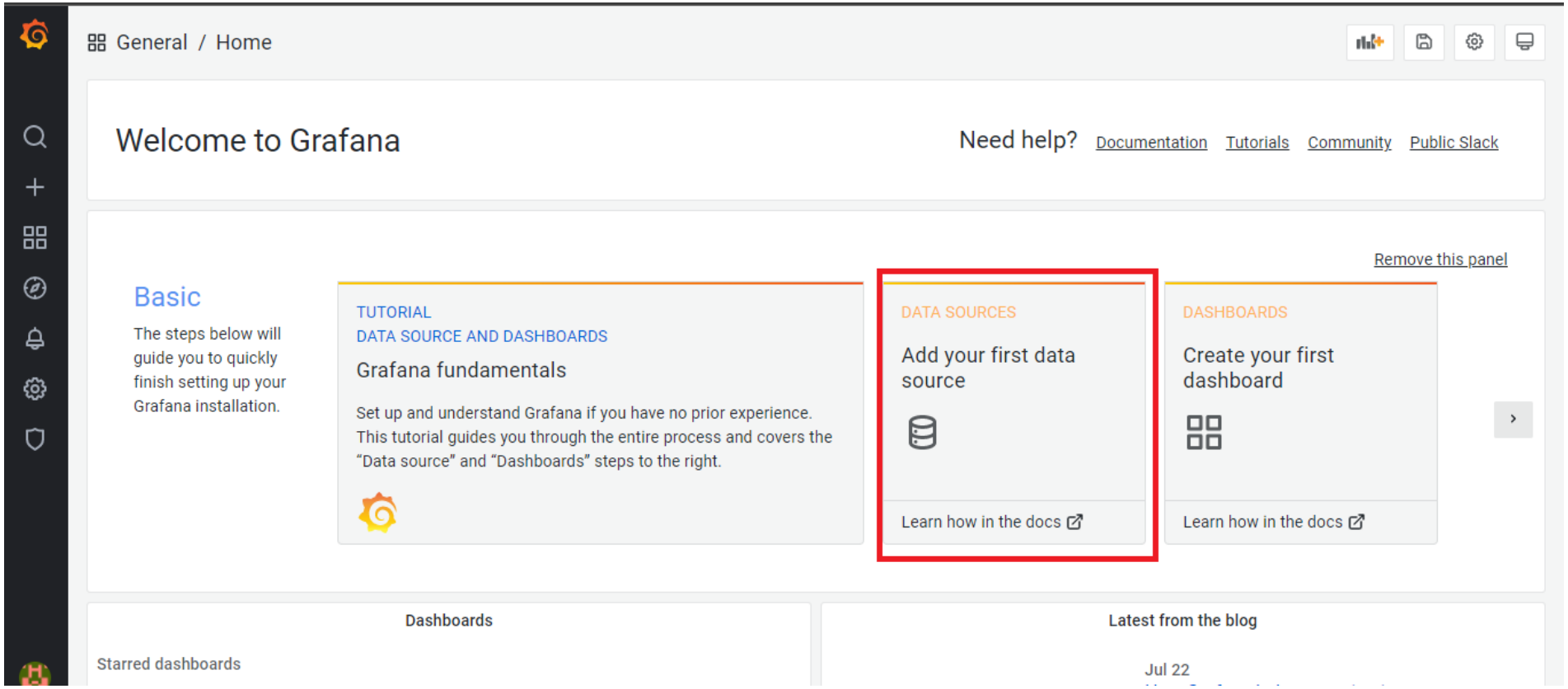
Grafana es un software de monitoreo de recursos mediante paneles gráficos, la solución que se propone también cuenta con un tablero en donde se pueden observar varias métricas de la blockchain.

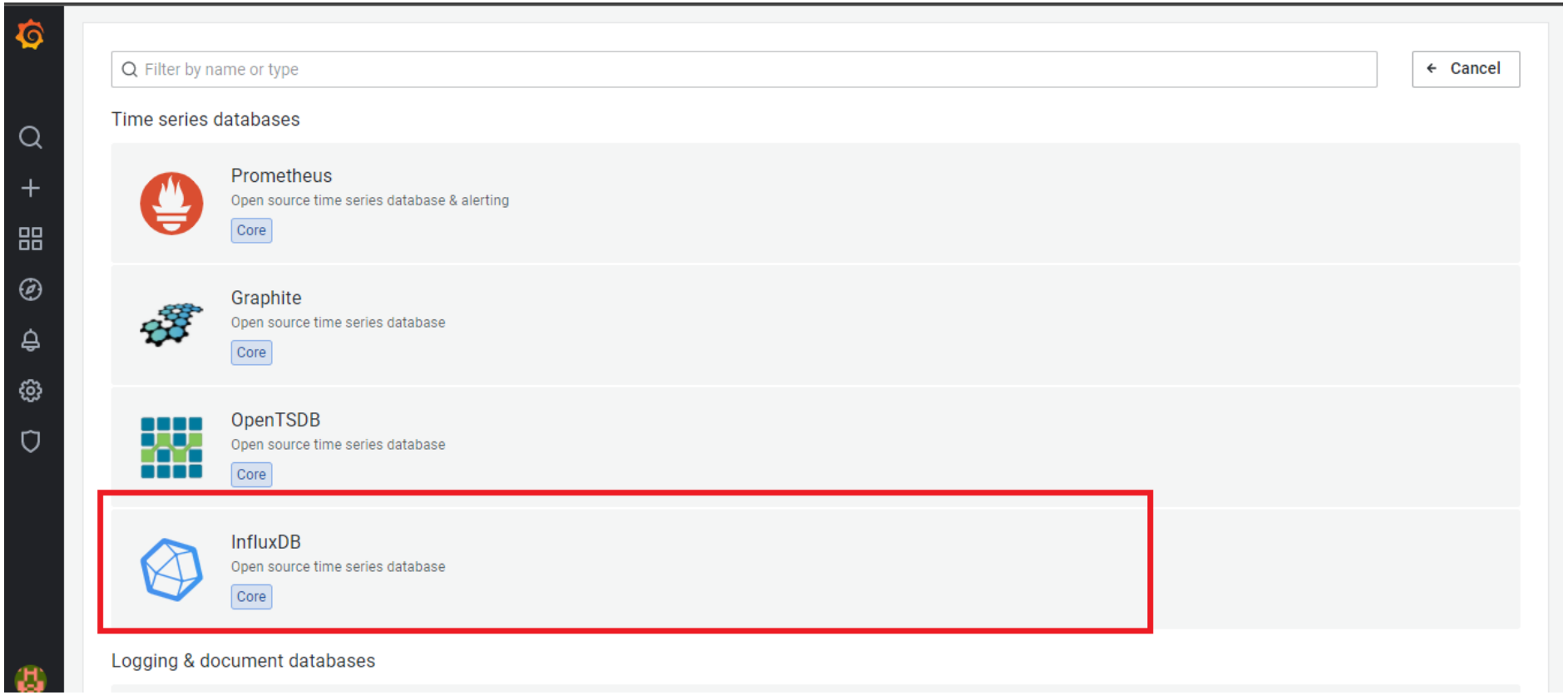
Para la configuración de Grafana debemos abrir en un navegador la ip del nodo 0 seguido del carácter dos puntos y el número 3000 que indica el puerto en el que se levanta Grafana.



La contraseña y clave predeterminada en “admin”.

Una vez iniciada la sesión se debe configurar la fuente de datos de **influxdb**, este proceso puede variar dependiendo de la versión de Grafana.


General / Home
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
DATA SOURCES
Add your first data source
Learn how in the docs
DASHBOARDS
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Latest from the blog
Jul 22


Filter by name or type
Cancel
Time series databases
Prometheus
Open source time series database & alerting
Core
Graphite
Open source time series database
Core
OpenTSDB
Open source time series database
Core
InfluxDB
Open source time series database
Core
Logging & document databases

HTTP

URL	http://localhost:8086
Access	Server (default) Help ›
Whitelisted Cookies	New tag (enter key to add)
Timeout	

Auth

Basic auth		With Credentials	
TLS Client Auth		With CA Cert	
Skip TLS Verify			
Forward OAuth Identity			

Custom HTTP Headers

+ Add header

InfluxDB Details

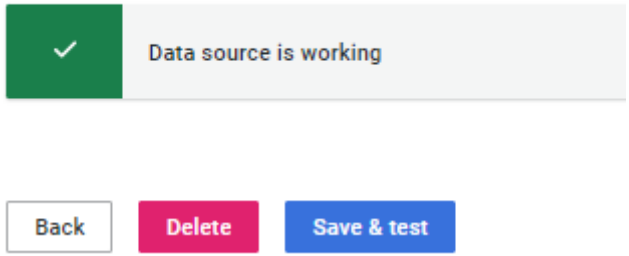
Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

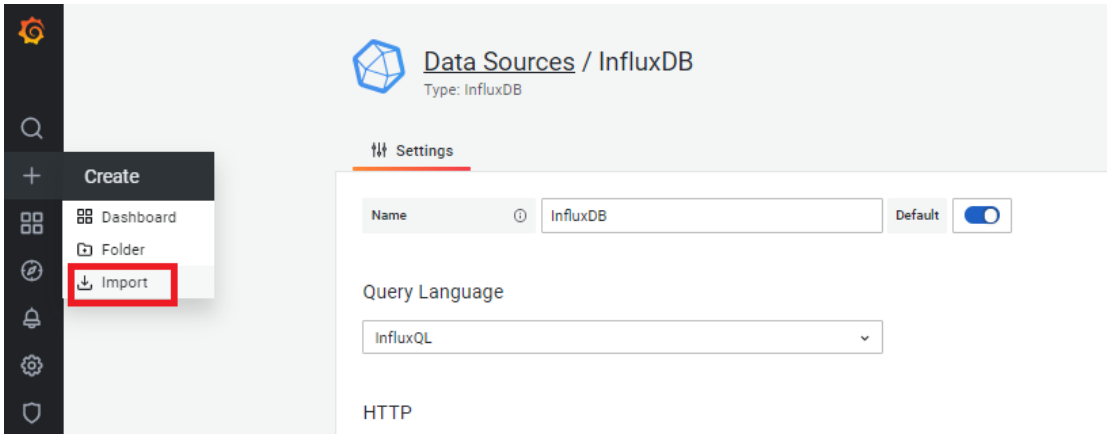
To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Database	metrics
User	lrdata
Password	•••••

En la **URL** se debe colocar “http://{ip del nodo 0}:8086”, en **Database** debe ir “metrics”, en **User** y **Password** debe ir “lrdata” (estos datos se configuran en los ficheros docker-compose-0.yml y docker-compose-0_0.yml dentro del directorio **docker** de la solución) y verificar que al guardar la conexión, esta funcione correctamente..



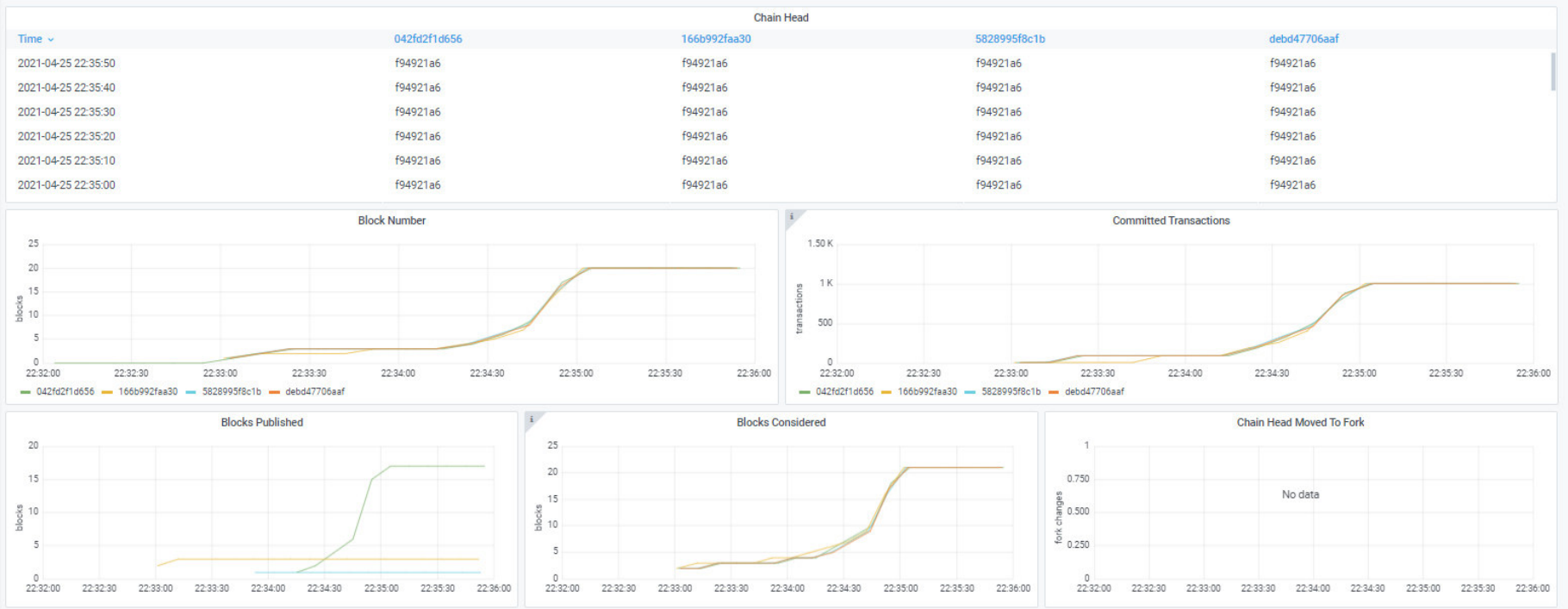
Una vez generada la fuente de datos, se debe importar el tablero que es un fichero .json y está en el directorio **./docker/grafana/dashboards** y apuntarlo hacia la fuente de datos creada.



Una vez realizado esto, podremos ver el tablero con las métricas de la blockchain (Anexo III)

Anexo III – Captura de pantalla del tablero de Grafana.

Captura de pantalla del tablero de Grafana luego de haber ejecutado una prueba de 1000 transacciones.



Anexo IV – Captura de pantalla de la aplicación frontend desarrollada.

Captura de pantalla de la aplicación frontend desarrollada en donde se pueden observar transacciones concurrentes en diferentes servidores de la base de datos. En este caso, el orden que refleja la blockchain se podría considerar el correcto.

Block	Transaction	Table	Application user	Data	Database server	Application time	Database time	Blockchain time
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
27	INSERT	transacciones	user6	{ "inserted": { "id": "3083", "id_cuenta_bancaria": "1", "id_tipo_transaccion": "2", "monto": "8.600000000000000e+003" } }	sqlserver3	2021-05-25 19:05:27.829	2021-05-25T19:05:27.920	2021-05-25 19:05:27.934
27	INSERT	transacciones	user10	{ "inserted": { "id": "3083", "id_cuenta_bancaria": "2", "id_tipo_transaccion": "2", "monto": "1.788000000000000e+003" } }	sqlserver0	2021-05-25 19:05:27.829	2021-05-25T19:05:27.913	2021-05-25 19:05:27.925
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
27	INSERT	transacciones	user8	{ "inserted": { "id": "3082", "id_cuenta_bancaria": "2", "id_tipo_transaccion": "2", "monto": "9.100000000000000e+003" } }	sqlserver1	2021-05-25 19:05:27.704	2021-05-25T19:05:27.793	2021-05-25 19:05:27.813
27	INSERT	transacciones	user5	{ "inserted": { "id": "3081", "id_cuenta_bancaria": "2", "id_tipo_transaccion": "1", "monto": "2.323000000000000e+003" } }	sqlserver0	2021-05-25 19:05:27.704	2021-05-25T19:05:27.783	2021-05-25 19:05:27.800