Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Tipo de artículo: Artículo original
Temática: Seguridad informática
Recibido: 30/06/2021 | Aceptado: 01/10/2021 | Publicado: 11/11/2021

Arquitectura distribuida de alta disponibilidad para la detección de fraude

High-availability distributed architecture for fraud detection

Alejandro García Núñez 0000-0002-5492-9427[1*]
Jorge Luis Olmedo Flores 0000-0003-2744-2733[2]

[1]Empresa de Telecomunicaciones de Cuba. Calle 56, No. 4306, E/ 43 y 45, apto 11, Playa, La Habana. alejandro.nunez@etecsa.cu
[2]Calle F No. 658 e/ 27 y 29, apto 2, Vedado, Plaza de la Revolución, La Habana. jorge.olmedo@etecsa.cu

[*]Autor para correspondencia: (alejandro.nunez@etecsa.cu)

RESUMEN

La detección temprana, rápida y eficaz del fraude en el sector de las telecomunicaciones se ha convertido en la punta de lanza para enfrentar las más complejas y diversas vías en la que pueden producirse los ataques y el fraude. Para su detección se emplean diferentes técnicas, herramientas y algoritmos como el aprendizaje automático el cual es una rama de la Inteligencia Artificial que permite a las computadoras aprender. Para poder aprovechar al máximo las ventajas del aprendizaje automático, se configuran arquitecturas de hardware y software robustas. Estas son configuradas de forma distribuida permitiendo a un conjunto de equipos trabajar como uno solo de forma transparente, aumentando el rendimiento y su procesamiento. El objetivo del presente trabajo es desarrollar una arquitectura distribuida de alta disponibilidad mediante la plataforma de datos Hortonworks que permita aplicar técnicas de aprendizaje automático en la detección de fraude. Se instalaron y configuraron los componentes de Apache que presenta como Spark, HBase y Hadoop los cuales permiten analizar tráfico en grandes cantidades de datos. Se muestra un ejemplo del resultado de aplicar el algoritmo de aprendizaje automático K-means empleando la librería PySpark para la creación de clusters. La instalación y configuración de la plataforma de datos Hortonworks dio como resultado una arquitectura que cuenta con

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

alta disponibilidad, flexible, escalable, tolerante a fallos y permite emplear el aprendizaje automático en la detección de fraude.

Palabras clave: Detección de fraude; Aprendizaje automático; Arquitectura distribuida.

ABSTRACT

Early, fast and effective fraud detection in the telecommunications sector has become the spearhead for dealing with the most complex and diverse ways in which attacks and fraud can occur. Different techniques, tools and algorithms are used for detection, such as machine learning, which is a branch of Artificial Intelligence that allows computers to learn. In order to take full advantage of the benefits of machine learning, robust hardware and software architectures are set up. These are configured in a distributed manner allowing a set of computers to work as one in a transparent way, increasing performance and processing. The objective of this work is to develop a highly available distributed architecture using the Hortonworks data platform that allows the application of machine learning techniques in fraud detection. Apache components such as Spark, HBase and Hadoop were installed and configured to analyze traffic in large amounts of data. An example of the result of applying the K-means machine learning algorithm using the PySpark library for the creation of clusters is shown. The installation and configuration of the Hortonworks data platform resulted in an architecture that is highly available, flexible, scalable, fault tolerant and allows the use of machine learning for fraud detection.

Keywords: Fraud detection; Distributed Architecture; Machine learning.

Introducción

Con el crecimiento que van alcanzando cada día más las vías existentes para comunicarse, se ha hecho imperioso observar y estudiar la información que manejan para la detección de fraude de una forma diferente. Una de estas formas es contar con sistemas que presenten en su diseño arquitecturas distribuidas. Un sistema distribuido es un conjunto de elementos informáticos autónomos que aparecen ante sus usuarios como un único sistema coherente. Esta definición hace referencia a dos rasgos característicos de los sistemas distribuidos: el primero es que un sistema distribuido es un conjunto de elementos informáticos, cada uno de los cuales puede

comportarse de forma independiente. Un elemento informático, al que generalmente nos referiremos como un nodo, puede ser un dispositivo de hardware o un proceso de software. Una segunda característica es que los usuarios (ya sean personas o aplicaciones) creen que están tratando con un único sistema. Esto significa que, de un modo u otro, los nodos autónomos deben colaborar. La forma de establecer esta colaboración es el núcleo del desarrollo de los sistemas de distribución. En principio, incluso dentro de un mismo sistema, podrían ser desde ordenadores centrales de alto rendimiento hasta peque nos dispositivos en redes de sensores (Maarten van Steen, 2018). Ejemplos de sistemas que emplean arquitecturas distribuidas se puede mencionar a Facebook, Twitter, el buscador de Google, Youtube, Gmail y sistemas en la nube entre otras. Algunas de las plataformas para la construcción de este tipo de arquitecturas se puede mencionar a Hortonworks la cual es basada en Hadoop y actualmente forma parte de Cloudera, la Plataforma de Datos de Cloudera (CDP) en la nube, DC/OS el cual es un sistema operativo distribuido basado en el núcleo de sistemas distribuidos Apache Mesos que permite el despliegue y orquestación de aplicaciones distribuidas en Docker entre otras.

La ciencia del dato es un término que en la actualidad tiene una importancia relevante para el hombre. En los últimos años, ha surgido como una disciplina nueva e importante. Puede ser visto como una amalgama de disciplinas clásicas como la estadística, la minería de datos, las bases de datos y los sistemas distribuidos. Los enfoques existentes deben combinarse para convertir los abundantes datos disponibles en valor para los individuos, las organizaciones y la sociedad. Además, han surgido nuevos desafíos, no sólo en términos de tamaño (”Big Data”), sino también en términos de las preguntas a responder (Van der Aalst, 2016). El análisis de los datos ha cobrado gran relevancia con el uso de Internet y las redes sociales, pues mientras más personas estén conectadas, se genera mayor cantidad de datos. En un reporte realizado por la empresa CISCO en Febrero de 2019, se identificó que el tráfico por cada teléfono inteligente en un mes durante 2017 era de 2Gb y ya para el 2022 se prevee un crecimiento de hasta 11GB (Cisco, 2019). Uno de los retos a que se enfrentan cotidianamente los especialistas, es el procesamiento de todo ese gran cúmulo de información y para ello se apoyan en el empleo de nuevas tecnologías como Big Data y procesamiento streaming, algoritmos para minería de datos o aprendizaje automático y sin lugar a dudas en las matemáticas. Muchas empresas del mundo realizan grandes inversiones y pagan enormes cantidades de dinero por los resultados obtenidos aplicando estas técnicas, pues gracias a ello pueden mejorar sus ofertas y atraer clientes. Las posibilidades de análisis son muchas y varían de acuerdo a la rama en que se esté realizando.

En las telecomunicaciones, la ciencia del dato se puede enfocar en el procesamiento del tráfico que genera el uso de Internet y servicios telefónicos como los mensajes de texto (SMS) y las llamadas. El estudio de esta

información puede derivar y ayudar en el reconocimiento de patrones de comportamiento, en la creación de perfiles de usuarios, en la detección de fraudes y ataques informáticos, en la automatización de procesos y por supuesto en evitar pérdidas monetarias. Para analizar el tráfico y que este a su vez pueda generar valor, en la actualidad una de las técnicas empleadas por los analistas de datos son los algoritmos de aprendizaje automático. Estos se clasifican en supervisados, semi-supervisados y no supervisados.

Las empresas de telecomunicaciones se ven expuestas a la ocurrencia de fraudes. La prevención y detección de fraudes es importante porque las empresas y los proveedores de servicios de telecomunicaciones pierden una proporción significativa de sus ingresos como resultado de ello (Hilas et al., 2015). Según la encuesta mundial sobre pérdidas por fraude del año 2017 realizada por la Communications Fraud Control Association (CFCA)[1], las empresas perdieron 29.2 billones de dólares (CFCA, 2018).

En el procesamiento del tráfico, los analistas se enfrentan a grandes retos y a diferentes problemas que limitan las capacidades de análisis como los que se mencionan a continuación:

- Las telecomunicaciones incluyen muchos servicios, cableados e inalámbricos, sistemas móviles y celulares, redes terrestres y satelitales y un gran grupo de aplicaciones para comunicarse a través de Internet haciendo más difícil la detección de fraude.
- Falta de conocimiento en nuevos tipos de fraude y preparación en este campo.
- El tráfico que se obtiene es demasiado lo cual provoca un aumento de la complejidad para detectar el fraude.
- Las herramientas con que cuentan se pueden ver limitadas en cuanto a la predicción de patrones, inferir conocimientos o no pueden aprender automáticamente.
- La toma de decisiones aplicando ciencia a los datos se puede convertir en una tarea compleja por la falta de conocimiento y al no uso de tecnologías y algoritmos enfocados en este aspecto.

El **objetivo general** del presente trabajo constituye: desarrollar una arquitectura distribuida de alta disponibilidad mediante la plataforma Hortonworks que permita aplicar técnicas de aprendizaje automático en la detección de fraude.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Métodos o Metodología Computacional

En el presente trabajo se realizaron estudios explicativos con el objetivo de conocer y explicar los conceptos relativos a las tecnologías a emplear en la confección de la arquitectura distribuida de alta disponibilidad. Esto dio la posibilidad de aplicar otros métodos como el descriptivo para poder describir características importantes de las tecnologías que son objeto de análisis y el exploratorio permitió la familiarización con conceptos desconocidos y obtener información.

Hortonworks

La plataforma de datos Hortonworks (HDP) se basa en Apache Hadoop para gestionar el almacenamiento, las consultas y el procesamiento. Tiene la ventaja de ser una solución rápida, rentable y escalable. Ofrece varios servicios de gestión, monitorización e integración de datos. Proporciona herramientas de gestión de código abierto y admite conexiones con algunas plataformas de BI (Business Intelligent). HDP garantiza un almacenamiento distribuido a través de DHFS y la base de datos no relacional, HBase. Permite un procesamiento de datos distribuido basado en MapReduce, la consulta de datos a través de Hue y la ejecución de scripts mediante Pig. HDP incluye Oozie para gestionar y programar flujos de trabajo, así como HCatalog para manejar servicios de metadatos. Hay muchas herramientas disponibles en HDP, incluyendo webHDFS, Sqoop, Ambari y Zookeeper. (Bhathal and Dhiman, 2018)

La imagen a continuación muestran la plataforma Hortonworks y los componentes que la conforman.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

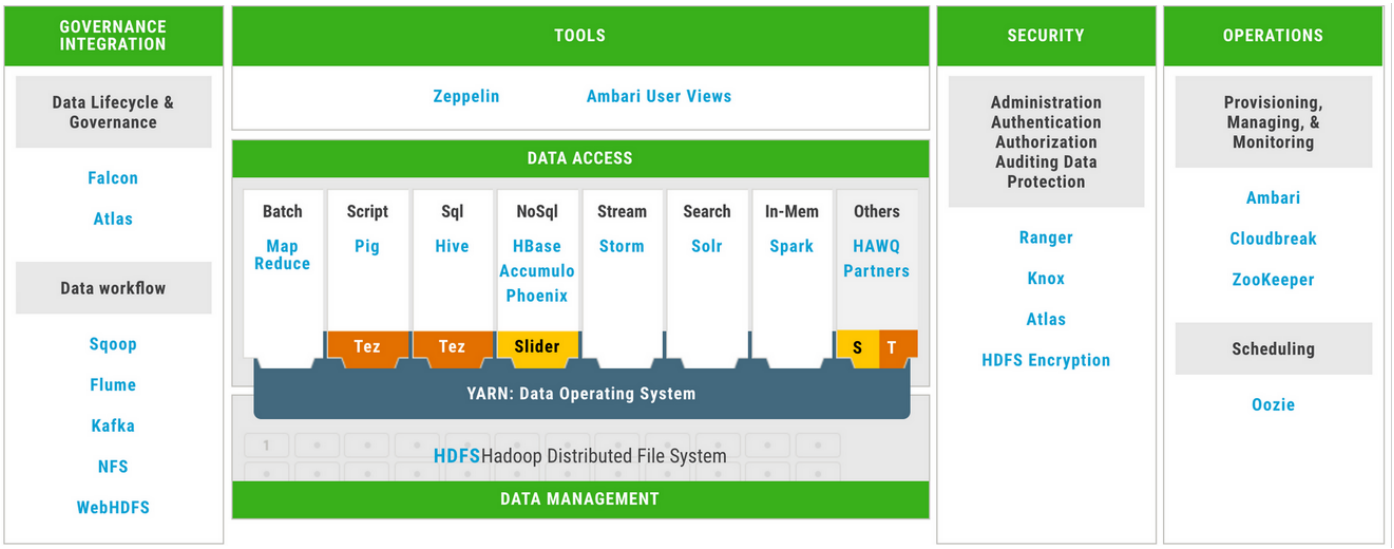

Fig. 1 - Plataforma Hortonworks y sus componentes.

https://community.cloudera.com/t5/Support-Questions/Difference-between-Apache-Hadoop-and-HDP/td-p/166751?

A continuación se explica como instalar y configurar una arquitectura en alta disponibilidad la cual mediante las bondades que brinda la plataforma, permitirá el escalado de los nodos de datos. Se explica como se elaboró la arquitectura y los componentes que la conforman los cuales pueden variar de acuerdo a las características del trabajo que se desea realizar. Algunos de estos componentes forman parte del ecosistema de Hadoop por lo cual al instalarlos, el software pedirá instalar otro debido a que es necesario para su funcionamiento.

Descripción de la arquitectura

Cloudera recomienda para los equipos cuando van a crear el cluster empleando Hortonworks, que comiencen con despliegues pequeños e ir escalando en correspondencia a las necesidades. De acuerdo a esa premisa se muestra a continuación los recursos para iniciar el cluster.

- **Sistema Operativo**

  Se empleó la distribución de Linux, Debian, en su versión 9.

- **Repositorios**

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Servidores	Cantidad	Memoria	Procesamiento	Almacenamiento
Ambari	1	4 GB	4	40
Zookeeper	3	4 GB	4	40
Hadoop, HBase, Yarn, MapReduce, Spark y Kafka (NameNode)	2	24 GB	16 - 24	500 GB
Hadoop, HBase, Yarn, MapReduce, Spark (DataNode)	3	16 GB	8	1 TB

Tabla 1 - Recursos asignados para la arquitectura de alta disponibilidad.

Se descargaron y configuraron los repositorios de Debian, Ambari y Hortonworks para poder acceder a los paquetes necesarios para la instalación del cluster y posteriormente los componentes de la plataforma.

- **Apache Ambari**

  Apache Ambari es una herramienta de administración de código abierto. Se despliega en clusters Hadoop y se utiliza para monitorear los programas que se ejecutan en el clúster. Apache Ambari es una aplicación web que controla, comprueba y proporciona un funcionamiento seguro de los clústeres Hadoop. (Minukhin et al., 2021)

  Para una mejor elección de los recursos de hardware en el servidor de Ambari puede emplear la siguiente tabla de acuerdo a la cantidad de nodos que desee configurar.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Cantidad de servidores	Memoria	Almacenamiento
1	1 GB	10 GB
10	1 GB	20 GB
50	2 GB	50 GB
100	4 GB	100 GB
300	4 GB	100 GB
500	8 GB	200 GB
1000	12 GB	200 GB
2000	16 GB	500 GB

Tabla 2 - Recursos a asignar al servidor de Ambari de acuerdo a la cantidad de nodos del cluster.

$https://docs.cloudera.com/HDPDocuments/Ambari-2,7,5,0/bk_ambari-installation/$

Instalación de Ambari

Ambari está basado en una arquitectura cliente servidor por lo cual para una correcta instalación es necesario instalar:

- **Servidor Ambari**

  Paquete que se instala solo en el servidor de Ambari. Será el encargado de controlar, manejar y monitorear el cluster. Recibe información de los nodos mediante el cliente de Ambari. El Servidor Ambari utiliza por defecto una base de datos PostgreSQL incrustada. Cuando se instala, los paquetes y dependencias de PostgreSQL deben estar disponibles para su instalación. Estos paquetes suelen formar parte de los repositorios del sistema operativo. Se puede instalar en una base de datos externa.

- **Cliente Ambari**

  Paquete que se instala en cada nodo del cluster. Envía información al Servidor Ambari del estado de los nodos. A través de este paquete es que se instalan los componentes del cluster.

- **Apache Zookeeper**

  Apache ZooKeeper es un servicio para mantener la información de configuración, nombrar y proporcionar servicios de sincronización distribuida y de grupo. Varios clientes pueden acceder a ZooKeeper simultáneamente. Las acciones pueden ser sincrónicas o asíncronas. Las acciones sincrónicas bloquean

(suspenden) el cliente activo hasta que el servidor devuelva el resultado directamente a ese hilo. Las acciones asíncronas se envían al servidor sin bloquear el hilo del cliente que llama. Implementa un almacén de datos distribuido de alto rendimiento. Puede operar como uno o más servidores, y cada servidor permite el acceso concurrente por múltiples clientes. Los datos se organizan como nodos en un árbol, similar a un sistema de archivos. (Artho et al., 2017)

Instalación de Apache Zookeeper

Se debe instalar en al menos tres nodos para garantizar la tolerancia a fallos. La cantidad de nodos puede variar en dependencia de los componentes a instalar. Siempre debe ser una cantidad impar para garantizar un correcto proceso de elección del líder en el cluster de Zookeeper. En el presente trabajo se seleccionó tres nodos para su instalación.

- **Hadoop, MapReduce, Yarn HBase, Spark y Kafka**

Apache Hadoop

Apache Hadoop es un software de código abierto que distribuye el procesamiento de grandes datos entre múltiples nodos o un solo nodo. El usuario de Hadoop tiene la capacidad de procesar el gran volumen de datos en múltiples nodos, lo que requiere la instalación de Hadoop como modelo pseudo-distribuido. Los componentes principales de Hadoop son reconocidos como Hadoop Distributed File System (HDFS) y MapReduce. El HDFS procesa el conjunto de datos utilizando el sistema NameNode y el sistema DataNode. El NameNode tiene la función principal de contener el archivo del conjunto de datos y otra información relacionada, como el nombre del archivo del conjunto de datos, el permiso y la ubicación de cada bloque del conjunto de datos. Por consiguiente, el NameNode está conectado al DataNode, donde se almacena una réplica de los bloques de datos en la memoria y se procesa para las operaciones de lectura y escritura. (Almansouri and Masmoudi, 2019)

MapReduce

MapReduce es un marco de software que se considera parte del marco Hadoop. Tiene la capacidad de procesar un gran volumen de datos en terabytes que están disponibles en varios miles de nodos. MapReduce aplica la función reductora sobre los objetos no deseados. La funcionalidad de mapeo y reducción se realiza con precisión en el modelo local, así como en el modelo pseudo-distribuido con la asociación de YARN (Yet Another Resource Negociator) (Almansouri and Masmoudi, 2019)

Yarn

El Yarn (Yet Another Resource Negotiator) es el sistema de gestión de recursos de cluster de Hadoop. La gestión de los recursos del clúster hace referencia a la memoria, el CPU, etc. Permite que múltiples aplicaciones se ejecuten al mismo instante en el mismo clúster compartido y permite que las aplicaciones negocien los recursos en función de sus necesidades. Por lo tanto, Yarn se convierte en un sistema operativo de datos para Hadoop 2, ya que permite que varias aplicaciones coexistan en el mismo clúster compartido. Yarn tomó el potencial de gestión de recursos del clúster para el sistema MapReduce, de modo que los nuevos motores pudieran utilizar este potencial de gestión de recursos del clúster general. Esto aligeró el sistema MapReduce para centrarse en la parte de procesamiento de datos, que es mejor y preferiblemente seguirá siéndolo.(Perwej et al., 2017)

Apache HBase

HBase es una base de datos distribuida, no SQL, construida sobre el sistema de archivos de Hadoop. Su principal característica es el almacenamiento orientado a columnas. Es una parte del ecosistema Hadoop que proporciona acceso aleatorio de datos para leer/escribir en tiempo real. Se pueden almacenar los datos en HDFS directamente o a través de HBase. Al igual que HDFS y MapReduce, HBase también sigue arquitectura maestro/esclavo. Es responsabilidad de HMaster (maestro) de asignar regiones a HRegionServers (esclavos) y de recuperarse de los fallos de HRegionServer. HRegionServer es responsable de gestionar las solicitudes de los clientes para la lectura y la escritura. HBase utiliza Zookeeper para la gestión del clúster HBase.(Pandagale and Surve, 2016)

Apache Spark

Spark es un importante motor de análisis consolidado para un amplio procesamiento de datos distribuidos y cargas de trabajo de aprendizaje automático. Ha establecido un amplio dominio para resolver problemas de ciencia de datos e ingeniería utilizando lenguajes de programación como Python. Spark refuerza técnicas como el procesamiento en memoria, el flujo y el procesamiento por lotes de cargas de trabajo de Big Data.(Shaikh et al., 2019)

Aunque Spark utiliza muchos principios similares al motor de Hadoop MapReduce, Spark supera a este último en términos de rendimiento. Por ejemplo, dada la misma carga de trabajo de procesamiento por lotes, Spark puede ser más rápido que Hadoop MapReduce debido a la **computación de cálculo en memoria** utilizada por Spark en comparación con la tradicional lectura y escritura en el disco utilizada por Hadoop MapReduce. Spark puede ejecutarse en modo autónomo o puede combinarse con Hadoop para sustituir al motor Hadoop MapReduce. (Shaikh et al., 2019) Spark es una plataforma de computación en clúster rápida, tolerante a fallos y de uso general para el procesamiento de big data.(Minukhin et al.,

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

2021)

A continuación se muestra el ecosistema de Apache Spark donde se puede observar los diferentes componentes que lo integran.

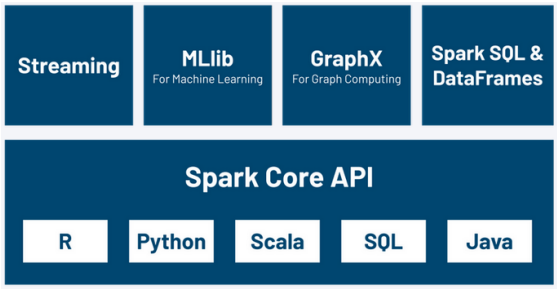

Fig. 2 - Ecosistema de Apache Spark

https://www.datamechanics.co/apache-spark/

- **MLib** Ofrece algoritmos de alta calidad con alta velocidad y hace que el aprendizaje automático sea fácil de usar y escalar. Varios algoritmos de aprendizaje automático como regresión, clasificación, agrupación en clústeres y álgebra lineal están presentes. También proporciona una biblioteca para el aprendizaje automático de primitivas de nivel inferior como el algoritmo de optimización genérica del descenso de gradiente. También proporciona otras funciones como el modelo evaluación e importación de datos. Se puede utilizar en Java, Scala, y Python. (Shaikh et al., 2019)

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Apache Kafka

Apache Kafka (Hiraman et al., 2018) es una plataforma para el entorno en tiempo real que utiliza un sistema de mensajería pública-suscrita distribuida y puede manejar un gran volumen de datos que le permite enviar mensajes en un punto final. Escala fácilmente sin tiempo de inactividad. Está diseñado para trabajar con un alto volumen de datos. Está particionado, replicado distribuido y con tolerancia a fallos. Prevee la ingestión del nuevo flujo de datos desde el productor. Para centrarse en la mensajería tradicional requiere una baja latencia. Está compuesto por las siguientes terminologías:

- **Tópico:** Un tópico es un sistema de alimentación a través del cual los mensajes se almacenan y publican y organizados en tópicos. Las aplicaciones productoras escriben datos en los tópicos y las aplicaciones consumidoras leen de los tópicos. Un tópico de Kafka está dividido en múltiples particiones.
- **Productores:** Los productores son los publicadores de mensajes a uno o más tópicos de Kafka. Los productores envían datos a los brokers de Kafka.
- **Consumidores:** Leen los datos de los brokers. Se suscriben a uno o más tópicos y consumen los mensajes publicados extrayendo datos de los brokers.
- **Conectores:** Se encargan de extraer los datos del flujo de los productores y entregar los datos del flujo a los consumidores o procesadores de flujo.
- **Procesador de flujos:** Los procesadores de flujos son aplicaciones que transforman los flujos de datos de los tópicos a otros flujos de datos de tópicos en Kafka Cluster.
- **Broker:** El clúster de Kafka suele estar formado por múltiples brokers para mantener el equilibrio de carga. Los brokers de Kafka no tiene estado, por lo que utilizan Zookeeper para mantener su estado del clúster.
- **Zookeeper:** Zookeeper se utiliza para gestionar y coordinar el broker de Kafka. El servicio Zookeeper se utiliza principalmente para notificar al productor y al consumidor la presencia de un nuevo broker en el sistema Kafka o fallo del broker en el sistema Kafka.

En la imagen a continuación se muestra un ejemplo práctico del funcionamiento de Apache Kafka.

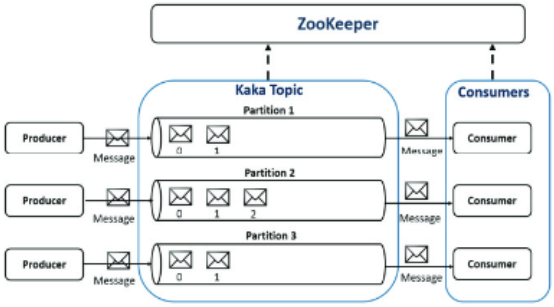

Fig. 3 - Funcionamiento Apache Kafka

https://ieeexplore.ieee.org/document/8533771

Instalación de Hadoop, MapReduce, Yarn HBase, Spark y Kafka

Los clusters Hadoop y HBase tienen dos tipos de máquinas: maestros y esclavos.

- Maestros: HDFS NameNode, YARN ResourceManager, y HBase Master.
- Esclavos: HDFS DataNodes, YARN NodeManagers, y HBase RegionServers.

En el proceso de instalación de esta aplicación, los HBase Master se instalan en los nodos maestros y los HBase RegionServers en los nodos esclavos del cluster con el almacenamiento en HDFS.

Se recomienda instalar en nodos separados los HBase Master y los NameNodes de Hadoop debido al procesamiento que deben realizar estos nodos. En la presente arquitectura se instalarán en los mismos servidores pues como se explicó anteriormente, el escalado de nodos se realizará en dependencia de los requerimientos y como se comporte el procesamiento en la arquitectura diseñada.

Los HBase Master utilizarán el cluster de Zookeeper configurado.

En el proceso de instalación de Spark, en los nodos master se instala el Spark2 History Server. En la opción de Asignar esclavos y clientes, seleccione los nodos clientes de Spark desde los que se pueden enviar trabajos de Spark a YARN (Cloudera, Inc., 2017).

Hortonworks recomienda separar los nodos maestro y esclavo según (Cloudera, Inc., 2019) porque:

- Las cargas de trabajo de tareas/aplicaciones en los nodos esclavos deben estar aisladas de los maestros.
- Los nodos esclavos se retiran con frecuencia para su mantenimiento.

La instalación de Apache Kafka consiste en selecionar uno de los servidores del cluster de Hortonworks para instalar el broker. No es necesario seleccionar un nodo como esclavo y tampoco un cliente. Requiere Zookeeper previamente instalado para su funcionamiento.

Alta disponibilidad para cluster de Hadoop

A continuación se explica la configuración del cluster de Hadoop según (Cloudera, 2018) en su documentación.

La función de alta disponibilidad de HDFS NameNode le permite ejecutar NameNodes redundantes en el mismo cluster en una configuración activa/pasiva con una espera en caliente. Esto elimina el NameNode como un potencial punto único de fallo (Single Point of Failure: SPOF) en un cluster HDFS.

Anteriormente, si un cluster tenía un único NameNode, y esa máquina o proceso dejaba de estar disponible, todo el cluster dejaría de estarlo hasta que el NameNode se reiniciara o se iniciara en otra máquina. Esta situación afectaba a la disponibilidad total del cluster HDFS de dos formas principales:

- En el caso de un evento no planificado, como la caída de una máquina, el cluster no estaría disponible hasta que un operador reiniciara el NameNode.
- Los eventos de mantenimiento planificados, como las actualizaciones de software o hardware en la máquina NameNode, darían lugar a períodos de inactividad del cluster.

HDFS NameNode HA evita esto facilitando una rápida conmutación por error al nuevo NameNode durante la caída de la máquina, o una conmutación por error iniciada por el administrador durante el mantenimiento planificado.

En un cluster de HA típico, se configuran dos máquinas distintas como NameNodes. En un cluster en funcionamiento, una de las máquinas NameNode está en estado activo y la otra en estado standby.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

El NameNode activo es responsable de todas las operaciones de los clientes en el cluster, mientras que el standby actúa como esclavo. La máquina en espera mantiene suficiente estado para proporcionar una rápida conmutación por error (si es necesario).

Para que el nodo en espera mantenga su estado sincronizado con el nodo activo, ambos nodos se comunican con un grupo de procesos separados llamados JournalNodes (JNs). Cuando el nodo activo realiza cualquier modificación del espacio de nombres, el nodo activo registra de forma duradera un registro de modificación en la mayoría de estos JNs. El nodo en espera lee las modificaciones de los JNs y observa continuamente los JNs en busca de cambios en el registro de modificaciones. Una vez que el nodo en espera observa las ediciones, las aplica a su propio espacio de nombres. Cuando se utiliza QJM [2], los JournalNodes actúan como almacenamiento compartido del registro de edición. En un evento de conmutación por error, el standby se asegura de haber leído todas las ediciones de los JounalNodes antes de promoverse a sí mismo al estado activo. (Este mecanismo asegura que el estado del espacio de nombres está totalmente sincronizado antes de que se complete una conmutación por error).

El NameNode secundario no es necesario en la configuración de HA porque el nodo en espera también realiza las tareas del NameNode secundario.

Para proporcionar una rápida conmutación por error, también es necesario que el nodo en espera tenga información actualizada sobre la ubicación de los bloques en su cluster. Para ello, los DataNodes se configuran con la ubicación de los dos NameNodes, y envían información sobre la ubicación de los bloques y los heartbeats a ambas máquinas NameNode

Es vital para el correcto funcionamiento de un cluster de HA que sólo uno de los NameNodes esté activo a la vez. En caso contrario, el estado del espacio de nombres podría divergir rápidamente entre las dos máquinas NameNode, lo que provocaría una posible pérdida de datos. Esta situación se denomina escenario de cerebro dividido (Split Brain).

Para evitar la situación de cerebro dividido, los JournalNodes sólo permiten que un NameNode sea escritor a la vez. Durante la conmutación por error, el NameNode elegido para pasar a ser activo asume la función de escribir en los JournalNodes. Este proceso evita que el otro NameNode continúe en estado activo y, por lo tanto, permite que el nuevo nodo activo continúe con la conmutación por error de forma segura.

Integración de los componentes de la arquitectura

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Integración de HBase con HDFS

Como se mencionó anteriormente, HBase mejora los beneficios de HDFS con la capacidad de servir lecturas y escrituras aleatorias a muchos usuarios o aplicaciones en tiempo real. Su integración con HDFS da la posibilidad de aprovechar las ventajas de este en cuanto al procesamiento de datos distribuidos, la fiabilidad de los datos y el alto rendimiento del análisis masivo de datos mediante MapReduce. Por lo tanto HBase debe conocer como acceder a los datos que se encuentran en HDFS, para ellos en la configuración de HBase se debe editar el fichero hbase-site.xml y agregar las siguientes propiedades mencionadas en (Apache HBase Team, 2021):

1. Indicarle a HBase que funcionará en modo distribuido.
   <property>
   <name>hbase.cluster.distributed </name>
   <value>true </value>
   </property>
2. Configurar el acceso a los datos de HBase hacia la instancia HDFS.
   <property>
   <name>hbase.rootdir </name>
   <value>hdfs://localhost:8020/hbase </value>
   </property>

Integración de Spark con HBase

Spark se puede configurar para leer y escribir en la base de datos HBase, para ello es necesario usar un conector. Hortonworks proporciona dos conectores diferentes como explica en su documentación:

- Conector Hortonworks Spark-HBase
  Cuando se utiliza el conector desarrollado por Hortonworks, el contexto subyacente es el DataFrame, con soporte para optimizaciones como la poda de particiones, la inserción de predicados y la exploración. El conector optimiza los filtros a el nivel de HBase, acelerando la carga de trabajo. La contrapartida es una flexibilidad limitada, ya que hay que especificar el esquema por adelantado. El conector aprovecha la API estándar de Spark DataSource para la optimización de las consultas.(Cloudera, 2017)

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

- Conector Spark-HBase (SHC)
  Conector de Apache HBase que utiliza Conjuntos de Datos Distribuidos Resistentes (RDD: Resilient Distributed Dataset). Aprovecha las técnicas de optimización del procesamiento de datos relacionales sobre el almacén de valores clave distribuido y orientado a columnas (es decir, HBase). En comparación con los sistemas existentes, SHC realiza dos importantes contribuciones. En primer lugar, SHC ofrece una integración mucho más estrecha entre las optimizaciones del procesamiento de datos relacionales y el almacén de datos no relacionales, a través de una implementación plug-in que se integra con Spark SQL, un motor de computación distribuida en memoria para datos relacionales. El diseño hace que el mantenimiento del sistema sea relativamente fácil, y permite a los usuarios realizar análisis de datos complejos sobre el almacén de valores clave. En segundo lugar, SHC aprovecha el motor Spark SQL Catalyst para la optimización y el procesamiento de consultas de alto rendimiento, por ejemplo, la poda de particiones de datos, la poda de columnas, el pushdown de predicados y la localización de datos.(Yang et al., 2018)

Los dos conectores están diseñados para satisfacer las necesidades de diferentes cargas de trabajo. En general, utilice el Hortonworks Spark-HBase Connector para SparkSQL, DataFrame y otras cargas de trabajo de esquema fijo. Utilice el conector Spark-HBase basado en RDD para RDD y otras cargas de trabajo de esquema flexible.

Integración de Apache Spark con Apache Kafka

Apache Kafka es el responsable de ingestar los datos hacia la arquitectura donde Spark con las ventajas que brinda en cuanto al trabajo en memoria se encarga de realizar el procesamiento del volumen de datos recibidos.

A continuación se muestra una imagen donde se observa la integración de los componentes de la arquitectura.

¿Cómo se realiza la detección de fraude?

A nivel general consiste en la realización de un conjunto de pasos cíclicos que a la postre dan un resultado. El primer paso comienza con identificar la problemática, qué está ocurriendo, qué es lo que se desea resolver. En el segundo paso se recogen y almacenan los datos que pueden llegar desde diversas fuentes ya sea por lotes o en tiempo real mediante aplicaciones como Spark. Una vez obtenidos los datos, en el tercero se analizan y modelan. Este es el punto más complejo y puede llevar mucho tiempo ajustarlos, pues requiere crear un juego de datos que satisfaga las necesidades del negocio y que los algoritmos pueden extraer información relevante

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

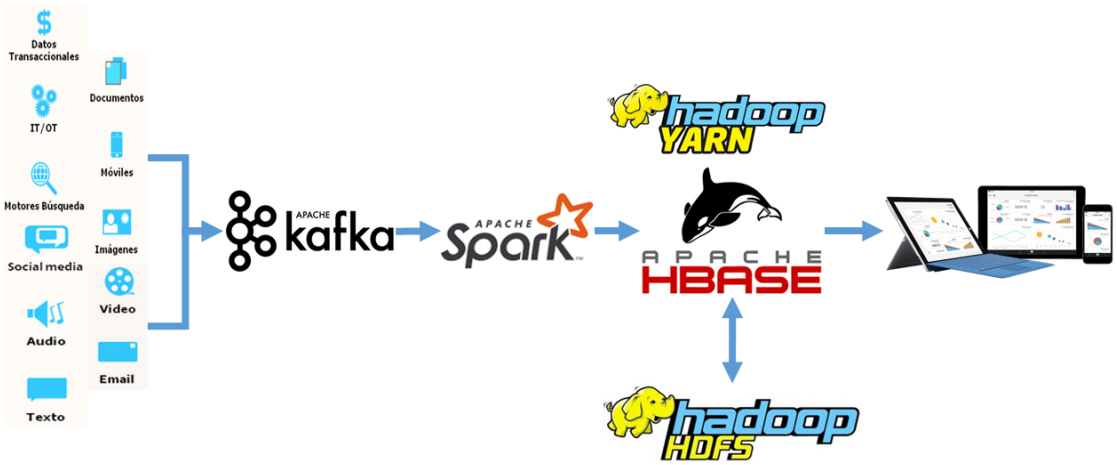

Fig. 4 - Integración de los componentes de la arquitectura

de ellos. El cuarto paso se trata de la visualización de los datos mediante reportes o gráficas. El quinto, con los resultados obtenidos, consiste en transformar el negocio de acuerdo a la información que se va obteniendo de los algoritmos implementados.

En el tercer paso es donde se insertan y aplican un conjunto de algoritmos de aprendizaje automático para la detección de fraude. El aprendizaje automático es el estudio de los algoritmos informáticos que mejoran automáticamente a través de la experiencia (Tom Mitchell, 1997).Es un campo de la Inteligencia Artificial (IA) en el que se investiga cómo los agentes informáticos pueden mejorar su percepción, cognición y acción con la experiencia. El aprendizaje automático consiste en que las máquinas mejoren a partir de los datos, el conocimiento, la experiencia y la interacción (Manuela Veloso, 2020).

PySpark

PySpark extiende el tiempo de ejecución de Spark para permitir la ejecución de programas Python sobre Spark. Normalmente, se envía un trabajo de PySpark (un proceso de Python), y se inicia una JVM para comunicarse con el proceso de Python), utilizando Py4J El proceso Python crea un objeto SparkContext en la JVM y el SparkContext orquesta el cálculo computación como el marco regular de Spark (para reutilizar casi la misma infraestructura de Spark). Sin embargo, la diferencia es que en PySpark, los ejecutores son procesos de Python. Cada proceso Python (por núcleo de CPU) se encarga de la ejecución de las tareas asignadas. Cada trabajador (programa basado en JVM) envía las tareas recibidas al conjunto de procesos Python y se comunica con ellos

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

mediante Pipes. El procesos de Python realizan el cálculo y almacenan los datos resultantes en un RDD (como objetos pickle) en la JVM. (Quoc et al., 2019). Mediante PySpark se pueden aplicar diferentes técnicas para la detección de fraude. Una de las más empleadas es el agrupamiento K-means. La idea principal del algoritmo k-means es que para obtener *k* clusters, el algoritmo selecciona *k* objetos de los conjuntos de datos. Los objetos restantes se asignan al objeto representativo más cercano, ya que el algoritmo intenta minimizar la distancia media al cuadrado entre los objetos. (Hilas et al., 2015).

A continuación se muestra un ejemplo de como queda la conformación de los clusters.

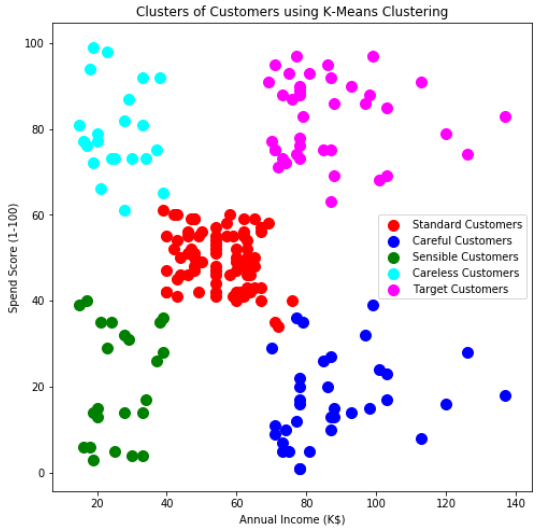

Fig. 5 - Ejemplo del algoritmo k-means con cinco clusters

https://analyticsindiamag.com/comparison-of-k-means-hierarchical- clustering-in-customer-segmentation/

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

La ejecución del algoritmo K-means como se puede observar en la imagen anterior, muestra 5 clusters donde los usuarios son clasificados de acuerdo a su comportamiento en el tráfico analizado. Una vez obtenido este resultado se pueden realizar otros estudios sobre la información obtenida aplicando otras técnicas de aprendizaje automático.

Resultados y discusión

El auge del BigData y el aprendizaje automático juegan un papel significativo en la reducción pérdidas debido al fraude en las telecomunicaciones. No solo se trata de las tecnologías en si mismas, si no de aplicar ciencias a los datos haciendo uso de ellas y ahí es donde los analistas y científicos de datos juegan un rol muy valioso. El empleo de técnicas de aprendizaje automático como el agrupamiento K-means analiza el tráfico de los usuarios y permite clasificar su comportamiento lo cual es muy importante en la detección de fraude.

Hortonworks es una plataforma con una amplia gama de tecnologías que posibilitan procesar y analizar un gran cúmulo de datos, las cuales pueden ser empleadas en la detección de fraude. Permite a los administradores desplegar, administrar, monitorear, garantizar la seguridad, acceso y gestión de los datos, la creación de flujos y escalar horizontalmente la arquitectura desplegada en caso necesario.

Como resultado del despliegue explicado en la sección anterior, en la siguiente imagen se puede apreciar la configuración en alta disponibilidad.

La solución de alta disponibilidad presentada posibilita que al encontrase uno de los NameNode del cluster fuera de servicio, la arquitectura es capaz de continuar su funcionamiento sin afectar el acceso a los datos.

La tolerancia a fallos de la presente arquitectura se garantiza mediante la sincronización que existe entre el NameNode y el StandbyNode mediante un grupo de nodos o procesos llamados JournalNodes.

En caso de un fallo del NameNode, el StandbyNode se asegura de que la información de sus metadatos se encuentre actualizada desde los JournalNodes antes de cambiar su rol al nuevo NameNode que permanecerá activo.

Si el NameNode y el StandbyNode fallan, el sistema queda fuera de servicio por lo cual es necesario restaurar lo antes posible el NameNode.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Mes Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-224

Fig. 6 - Configuración en alta disponibilidad de Hadoop

https://pt.slideshare.net/hortonworks/ambari-meetup/6

Mediante la herramienta Ambari se disminuyen las complejidades en cuanto a la gestión de Hadoop. Su interfaz Web, automatiza muchas de las operaciones del cluster como por ejemplo: la instalación, configuración, visibilidad estado del cluster y su propia gestión. Brinda la posibilidad de configurar alertas que ya están predefinidas en el sistema y de capturar y visualizar métricas mediante la herramienta Grafana.

La arquitectura de ZooKeeper ofrece alto rendimiento y disponibilidad con una baja latencia, pero el tamaño de la base de datos que ZooKeeper puede gestionar está limitado por la memoria de los nodos del cluster. Al contar con una configuración de nodos impar se garantiza que se lleve a cabo un proceso de elección del próximo líder en caso del fallo de algún nodo garantizando la alta disponibilidad y la tolerancia a fallos. Con el tipo de configuración seleccionado solo puede fallar un nodo, si se configuran cinco, es tolerante al fallo de dos servidores.

En cuanto al procesamiento, la presente arquitectura permite que los sistemas o aplicaciones existentes se puedan integrar con herramientas del ecosistema de software de Hadoop para realizar cargas masivas o de flujo. Permite ejecutar transformaciones utilizando múltiples opciones de acceso a los datos para el procesamiento por lotes, en memoria de forma más rápida con Apache Spark y en flujo a medida que llegan al cluster mediante Spark Streaming.

El trabajo en conjunto de Hadoop y Spark brinda la posibilidad de solventar algunas de las debilidades que presentan estas tecnologías lo cual posibilita en la presente arquitectura mejorar el rendimiento. En Spark el procesamiento se lleva a cabo en memoria lo cual es más rápido que realizarlo en disco como Hadoop, pero este no cuenta con un sistema de ficheros distribuido, por lo cual se puede integrar con Hadoop HDFS.

Con la instalación y configuración de Apache Hbase se mejora el trabajo de Hadoop HDFS mediante lecturas y escrituras rápidas y aleatorias en los datos almacenados. También se puede integrar con Apache Spark Streaming para la creación de flujos de datos en tiempo real.

Conclusiones

Con los resultados expuestos en la presente investigación se puede concluir que la solución presentada dio cumplimiento al objetivo general de desarrollar una arquitectura distribuida de alta disponibilidad para la detección de fraude. Para ello se instaló y configuró la plataforma Hortonworks la cual provee un conjunto de componentes como Apache Spark, Apache HBase y Apache Hadoop los cuales al integrarlos dan la posibilidad de emplear algoritmos de aprendizaje automático en la detección de fraude.

El empleo de técnicas de agrupamiento como el algoritmo K-means no debe ser vista como la única solución o la más factible. Es necesario aplicar en conjunto otras técnicas que validen ciertamente el comportamiento de los usuarios para decidir si están cometiendo fraude o no. La eficacia de los algoritmos a emplear dependen de cuan bien estén confeccionados los datos a analizar.

Futuros trabajos incluyen el estudio de otras tecnologías presentes en Hortonworks y que puedan ser usadas en la detección de fraude como es el caso de Apache Kafka. La creación de perfiles de usuarios es otro punto a estudiar en el futuro pues perfilar el comportamiento de los usuarios mediante el tráfico procesado es de gran importancia. Así como continuar profundizando en el estudio de otros algoritmos de aprendizaje automático.

Referencias

Hatim Talal Almansouri and Youssef Masmoudi. Hadoop distributed file system for big data analysis. In *2019 4th World Conference on Complex Systems (WCCS)*, pages 1–5, 2019. doi: 10.1109/ICoCS.2019.8930804.

Apache HBase Team. Apache HBase Reference Guide. https://hbase.apache.org/book.html, April 2021.

Cyrille Artho, Quentin Gros, Guillaume Rousset, Kazuaki Banzai, Lei Ma, Takashi Kitamura, Masami Hagiya, Yoshinori Tanabe, and Mitsuharu Yamamoto. Model-based api testing of apache zookeeper. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 288–298, 2017. doi: 10.1109/ICST.2017.33.

Gurjit singh Bhathal and Amardeep Singh Dhiman. Big data solution: Improvised distributions framework of hadoop. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 35–38, 2018. doi: 10.1109/ICCONS.2018.8663142.

CFCA. 2017 Global Fraud Loss Survey. http://v2.itweb.co.za/whitepaper/Amdocs{_}LINKED{_}2017{_}CFCA{_}Global{_}Fraud{_}Loss{_}Survey.pdf, 2018.

Cisco. Mobile Visual Networking Index (VNI) Infographic - Cisco. https://www.cisco.com/c/m/en{_}us/solutions/service-provider/visual-networking-index.html, 2019.

Inc. Cloudera. Hortonworks data platform apache spark component guide. https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.4/bk_spark-component-guide/content/ch08s05.html, December 2017.

Inc. Cloudera. Hortonworks data platform apache hadoop high availability. https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.5/bk_hadoop-high-availability/content/ha-nn-architecture.html, May 2018.

Cloudera, Inc. Hortonworks Data Platform Apache Spark Component Guide. https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.6.2/bk_spark-component-guide/bk_spark-component-guide.pdf, August 2017.

Cloudera, Inc. Planning for the HDP Cluster. https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/cluster-planning/content/typical-hadoop-cluster-hardware.html, August 2019.

Constantinos S Hilas, Paris Mastorocostas, Paris A Mastorocostas, and Ioannis T Rekanos. Clustering of telecommunications user profiles for fraud detection and security enhancement in large corporate networks:

a case study. https://www.researchgate.net/publication/274570263{%}0Ahttp://dx.doi.org/10.12785/amis/090407, 2015. ISSN 09067590.

Bhole Rahul Hiraman, Chapte Viresh M., and Karve Abhijeet C. A study of apache kafka in big data stream processing. In *2018 International Conference on Information , Communication, Engineering and Technology (ICICET)*, pages 1–3, 2018. doi: 10.1109/ICICET.2018.8533771.

Andrew S. Tanenbaum Maarten van Steen. Distributed systems. https://www.academia.edu/40661214/Distributed_Systems_3_, December 2018.

Herbert A. Simon Manuela Veloso. Aprendizaje Automático. https://www.ml.cmu.edu/, 2020.

Sergii Minukhin, Natalia Brynza, and Dmytro Sitnikov. Analyzing performance of apache spark mllib with multinode clusters on azure hdinsight: Spark-perf case study. In Sergii Babichev, Volodymyr Lytvynenko, Waldemar Wójcik, and Svetlana Vyshemyrskaya, editors, *Lecture Notes in Computational Intelligence and Decision Making*, pages 114–134, Cham, 2021. Springer International Publishing. ISBN 978-3-030-54215-3. doi: 10.1007/978-3-030-54215-3_8.

Ashwini A. Pandagale and Anil R. Surve. Hadoop-hbase for finding association rules using apriori mapreduce algorithm. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 795–798, 2016. doi: 10.1109/RTEICT.2016.7807935.

Dr. Yusuf Perwej, Bedine Kerim, Mohammed Adrees, and Osama Sheta. An empirical exploration of the yarn in big data. *International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 , Foundation of Computer Science FCS, New York, USA*, Volume 12:Page 19– 29, 12 2017. doi: 10.5120/ijais2017451730.

Do Le Quoc, Franz Gregor, Jatinder Singh, and Christof Fetzer. Sgx-pyspark: Secure distributed data analytics. In *WWW '19: The World Wide Web Conference*, pages 3563–3564. ACM Digital Library, 2019. doi: https://doi.org/10.1145/3308558.3314129.

Eman Shaikh, Iman Mohiuddin, Yasmeen Alufaisan, and Irum Nahvi. Apache spark: A big data processing engine. In *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pages 1–6, 2019. doi: 10.1109/MENACOMM46666.2019.8988541.

McGraw Hill Tom Mitchell. Machine learning. http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/mlbook.html, 1997.

Wil Van der Aalst. Process mining: Data science in action. https://rd.springer.com/chapter/10.1007/978-3-662-49851-4{_}1, 2016.

Weiqing Yang, Mingjie Tang, Yongyang Yu, Yanbo Liang, and Bikas Saha. Shc: Distributed query processing for non-relational data store. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1465–1476, 2018. doi: 10.1109/ICDE.2018.00165.

Conflicto de interés

No aplica

Contribuciones de los autores

1. Conceptualización: Alejandro García Núñez
2. Curación de datos: Alejandro García Núñez
3. Análisis formal: Alejandro García Núñez
4. Adquisición de fondos: N/A
5. Investigación: Alejandro García Núñez
6. Metodología: Alejandro García Núñez
7. Administración del proyecto: Alejandro García Núñez
8. Recursos: Alejandro García Núñez
9. Software: Alejandro García Núñez
10. Supervisión: Jorge Luis Olmedo Flores
11. Validación: Jorge Luis Olmedo Flores
12. Visualización: Jorge Luis Olmedo Flores

13. Redacción ? borrador original: Alejandro García Núñez

14. Redacción ? revisión y edición: Alejandro García Núñez