UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

Diseño e Implementación de Plataforma Big Data para la Detección de Anomalías

Reporte Técnico de Investigación presentado por:

Adrián Hernández Rivas 182996

Requisito para la obtención del título de:

MAESTRO EN CÓMPUTO APLICADO

Dr.rer.nat. Juan Carlos Acosta Guadarrama

Dr. Victor Manuel Morales Rocha

Ciudad Juárez, Chihuahua 19 de agosto de 2021

Ciudad Juárez, Chihuahua, a 19 de Agosto de 2021

Asunto: Autorización de Impresión

C. Adrián Hernández Rivas

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del documento de Diseño e Implementación de Plataforma Big Data para la Detección de Anomalías , para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Maestro en Cómputo Aplicado.

Sin otro particular, reciba un cordial saludo.

Prof.Dr.rer.nat. Juan Carlos Acosta Guadarrama

Profesor Titular de Seminario de Titulación IV

Declaración de Originalidad

Yo, Adrián Hernández Rivas, declaro que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.

Adrián Hernández Rivas

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología CONACYT, la oportunidad de haberme brindado el apoyo como becario para poder dedicarme como estudiante de tiempo completo a cursar y concluir la presente Maestría en Cómputo Aplicado. Además, reconozco con especial afecto y gratitud el apoyo y aportaciones brindadas por el Dr. Juan Carlos Acosta Guadarrama en el proceso de guía, dirección y revisión de este reporte de investigación. De igual manera, por los consejos brindados, recomendaciones y experiencias compartidas. Otro agradecimiento especial es para el Dr. Víctor Manuel Morales Rocha, responsable del Laboratorio Nacional de Tecnologías de la Información LANTI y codirector de este trabajo, por permitirme el acceso a los recursos de infraestructura computacional para la realización del presente proyecto. Además, por su importante apoyo, guía, recomendaciones y aportación de su visión personal en la culminación del presente reporte. Finalmente, pero no menos importante, un agradecimiento especial al Mtro. Oscar Ruiz Hernández, por colaborar en la investigación y ejecución de este proyecto.

Dedicatoria

Ni en medio de una pandemia, dejaré de agradecer con máximo cariño y afecto el apoyo brindado de familia, amigos y profesores a quienes dedico este trabajo.

Índice general

Índice de figuras

Índice de tablas

Introducción

A inicios del 2016 el Foro Económico Mundial [1], presentó la iniciativa para establecer cómo las tecnologías de Inteligencia Artificial IA, Internet De las Cosas: también conocido como IoT (por sus siglas en inglés), Robótica Avanzada, Impresión 3D y Analítica de Datos enfocada a *Big Data* para la generación de valor de la información, han dado la pauta para una transformación al modo en que se generan y entregan distintos bienes y servicios. Particularmente, para múltiples industrias surgieron nuevas estrategias para cambiar las formas tradicionales en que se desarrollan actividades relacionadas a su actividad productiva, tales como: diseño, manufactura, monitoreo, control, ensamble y optimización de procesos. [2].

Sin embargo, para el uso y adopción de las tecnologías descritas en el párrafo anterior, existe una problemática subyacente relacionada al uso eficiente y administración de recursos e infraestructura requerido por las mismas. Es decir, para las organizaciones es importante mantener una relación de costo-beneficio [3].

En ese sentido, para el caso de las tecnologías relacionadas con el análisis masivo de datos y su uso a través del modelo de plataforma como servicio (PaaS, por sus siglas en inglés) se ha experimentado una constante evolución permitiendo una portabilidad anteriormente restringida en la migración de aplicaciones, sistemas de software y ecosistemas para el análisis de datos, entre ellos los de *Big Data* [3].

Soluciones de software y arquitecturas de sistemas se pueden construir a la medida y encapsular con la tecnología de contenedores. De ese modo, dicha flexibilidad representa

un potencial para el ahorro y gestión eficiente de recursos [4],[3]. Por ejemplo, según [3], una organización que tiene una aplicación de control de inventarios con almacenamiento intensivo de datos, podría gastar mensualmente $100,000 dolares USD (Dólar estadounidense, por sus siglas en inglés) con un proveedor de nube, por ejemplo, AWS (Amazon Web Services, por sus siglas en inglés), ahora puede migrar la misma aplicación con otra organización que le suministre el mismo servicio digital reduciendo costos por dicha prestación.

Existen otros beneficios del modelo de plataforma como servicio, tales como: implementación de software y prototipos que involucran una pila de tecnologías emergentes sin la preocupación de la configuración, despliegue y control de recursos de infraestructura, cómputo y almacenamiento [5]. En ese sentido, el desarrollador puede implementar y depurar aplicaciones y tecnologías de interés para el modelo de negocio que así lo requiera, pero de una forma más eficiente.

En este reporte de investigación se describe la implementación de un clúster para la ejecución de sistemas *Big data* a través de la modalidad plataforma como servicio. Conformado por nueve máquinas virtuales interconectadas a través de la red de área local de la UACJ en el Laboratorio Nacional de Tecnologías de la Información LANTI, sede Universidad Autónoma de Ciudad Juárez, UACJ.

El clúster opera utilizando un sistema operativo anfitrión de cómputo distribuido DCOS (Distributed Cloud Operating System, por sus siglas en inglés), que integra un esquema de computación de alto rendimiento con soporte nativo de contenedores Docker. De ese modo, se pueden incorporar los componentes relacionados con sistemas *Big Data* previamente encapsulados con esta tecnología y desplegarse en un modelo *multitenant*[1] .

[1] Según: https://ibm.co/3rqX0Pw. Es la arquitectura de software multiinquilino, que permite a varios usuarios compartir una sola instancia de una aplicación de software y sus recursos subyacentes

Capítulo 1

Planteamiento del Problema

Para lograr la transformación digital la organización, entidad, empresa , firma o industria debe integrar tecnologías de información y comunicación [2]. Por ejemplo, tecnologías actualmente extendidas, tales como: el uso de cómputo y servicios en la nube, infraestructura, almacenamiento, sistemas de software, *Big Data* y en algunos casos servicios de IA como *Machine Learning*[1] , por mencionar algunas. En ese sentido, la gestión eficiente de los recursos involucrados es prioritaria por dos razones: mantener sistemas en buen funcionamiento y controlar el aprovechamiento de sus recursos para obtener un máximo rendimiento (en lo posible) y minimizar el costo anual. En el presente capítulo se revisan los antecedentes que han impulsado la transformación digital. Además, también se presenta la descripción del problema, objetivo general, específicos, alcance, limitaciones y metodología.

1.1. Antecedentes

Transformación digital se refiere a los cambios en operaciones comerciales, productos, procesos, estructuras orgánicas y de gestión en una entidad u organización [2]. Uno de sus

[1] Los avances en el campo del reconocimiento de patrones y la aproximación de funciones para descubrir la relación entre las variables, es el área de la que se ocupa el aprendizaje automático, término extendido bajo el anglicismo de *Machine Learning*

objetivos es promover el ingreso de las industrias en un mercado cada vez más competitivo en el contexto de una economía globalizada y tiene cuatro dimensiones principales:

- Uso de tecnologías de la información. Aspecto que está directamente relacionado con la filosofía que adopta una empresa hacia las tecnologías de la información, así como su capacidad para explotar las mismas con el fin de obtener ventajas en el negocio.
- Cambios en la generación de valor. Productos o servicios se ven influenciados por la digitalización, cambiando la forma en que son entregados y monetizados, abriendo nuevos canales de venta y segmentos de mercado.
- Cambios estructurales. Se enfocan en transformar la estructura organizacional de una empresa, enfocándose en la posición de las nuevas actividades digitales dentro de las estructuras corporativas.
- Aspectos financieros. Para impulsar la transformación digital dentro de una empresa u organización debe existir una inversión económica solida por parte de las mismas.

En la Figura 1.1, tomada y modificada de [6], el aspecto central: Aspectos Financieros, es el elemento de mayor relevancia en el proceso de transformación digital, además de la implementación de cambios que propicien la generación de valor en una empresa, tal como el uso de tecnologías y la definición de nuevas estructuras organizacionales estrechamente relacionadas [6].

Big Data, es un término utilizado para describir la información que tiene las propiedades definidas por las tres "V" (Volumen, Variedad, Velocidad) [7]. *Volumen*, es un atributo que representa cantidades masivas de datos; *Variedad*, cualidad que indica la diversidad de los datos, es en relación a su estructuración interna; *Velocidad*, esta propiedad tiene en consideración una rápida tasa de transferencia de los datos. De ese modo, es importante disponer con una red informática capaz de soportar las transacciones realizadas en el tiempo mínimo

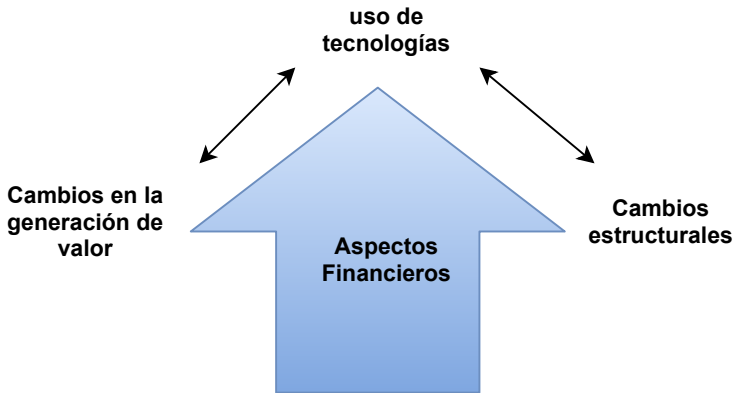


Figura 1.1: Dimensiones de la transformación digital.

posible. Según [8], una de las tecnologías que aportan valor a la transformación digital es *Big Data* y su generación de valor a través de analíticas avanzadas.

Las anomalías, según [9], se pueden comprender a través del análisis de diferentes tendencias de comportamiento en el tiempo. Por ejemplo, dos generadores eólicos manufacturados por un mismo proveedor que fueron habilitados y puestos en funcionamiento de forma sincronizada en el mismo campo de cultivo, tienden a desarrollar comportamientos y tendencias diferentes a lo largo del tiempo, debido a variables como uso, mantenimiento, temperatura, humedad, radiación solar, reparaciones, clima y desgaste general del equipo. De ese modo, cada variable tiene parámetros de tolerancia como indicador de buen funcionamiento. Cuando se registran valores fuera de este rango es cuando se presentan las anomalías, tales como: mal funcionamiento, desgaste mayor del producto o equipo, constantes fallas, errores de difícil detección.

Desde esa perspectiva, una solución viable a dicho problema es la utilización de nuevas tecnologías y componentes orientados a la implementación del cómputo de alto rendimiento usando hardware genérico y de bajo costo interconectado por una red informática. De ese modo, surge un nuevo paradigma de diseño e implementación de arquitecturas elásticas y escalables en una organización favoreciendo el uso de Inteligencia Artificial y *Big Data* [8].

Los datos en la última década han registrado un crecimiento exponencial y son originadas por diversas fuentes conectadas a Internet [10], tales como dispositivos inteligentes, sensores y software. Sin embargo, las diversidad de formato y origen de la información trae consigo nuevos desafíos con respecto a su procesamiento, incluyendo el análisis y la integración de implementaciones *Big Data* como tecnología de apoyo en la toma de decisiones.

En ese mismo sentido, obtener valor agregado de la información y nuevos significados es el enfoque primario de *Big Data* [8]. Además, con las capacidades tecnológicas actuales, es posible la adquisición de datos y análisis en tiempo real, cambiando el panorama de inversión de diversas industrias de un estilo conservador a uno cuantitativo y basado en modelos de predicción. Por otro lado, es importante mencionar las tendencias que actualmente promueven la revolución del *Big Data*. Según [8], podemos mencionar la siguientes:

El incremento acelerado de los datos

La información generada y adquirida en la actualidad tiene un patrón de crecimiento exponencial. Según el reporte de [8], aproximadamente el 90 % de los datos en el 2017 fueron generados en solo tres años. Una predicción en este mismo reporte estima 44 Zettabytes (1,000 millones de Terabytes) para 2024.

Reducción de costos en cómputo y almacenamiento

Las ventajas de la computación de alto rendimiento, así como el aumento en la disponibilidad del almacenamiento en la nube y acceso compartido de dichos recursos a través de Internet es otro factor importante para extender el uso del *Big Data*. La Figura 1.2, tomada de [11], es un estudio realizado a 750 organizaciones en Estados Unidos, muestra que el 20 % de las empresas informaron un gasto anual en uso de tecnologías de nube excedente a los doce millones de dólares y 74 % supera los 1.2 millones al año. El 26 % en el rango de 600,000 a 1.2 millones USD. Los *frameworks* de código abierto para el cómputo que se ejecuta de forma distribuida en clúster (es decir, dividir una tarea compleja en segmentos de tarea más pequeños y repartirlos en equipos de

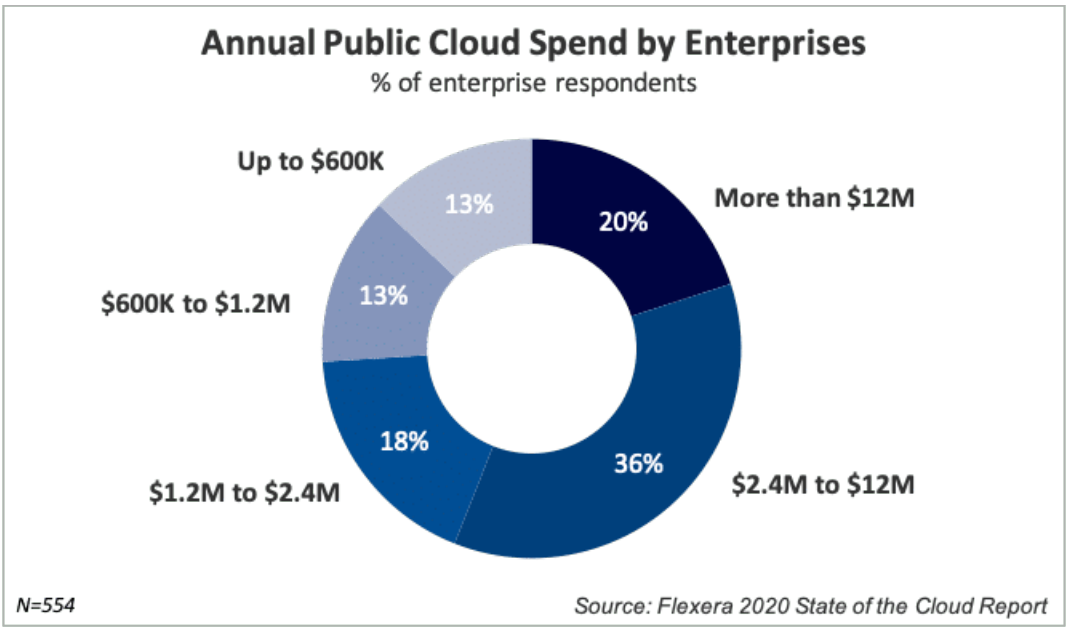


Figura 1.2: Reporte anual de gastos en nube pública, Flexera 2020.

cómputo que entregarán un resultado) como Apache Spark, han ganado terreno como principales componentes del *Big Data* [12]. Además, los proveedores de tecnología en nube brindan acceso remoto a través de sus PaaS. De tal manera que, este último factor ha desvanecido drásticamente las barreras y obstáculos iniciales para llevar a cabo el procesamiento y análisis de datos a gran escala. De esa manera, se generan estrategias que impulsan la toma de decisiones, basadas en los análisis de grandes volúmenes de información [8].

Avances en los métodos de *Machine Learning*

A través de nuevos métodos de ML se puede extraer valor de *Big Data* de maneras cada vez más complejas y efectivas, con ellos es posible construir sistemas de predicción y dar soporte a la toma de decisiones en una entidad u organización que haga uso de los mismos.

PaaS y *Big Data*

El cómputo en la nube ofrece los siguientes tres modelos de servicio: infraestructura como servicio (IaaS, por sus siglas en inglés), pone a disposición del usuario hardware y cómputo, por ejemplo: máquinas virtuales (vm, por sus siglas en inglés), bases de datos y almacenamiento. Además, software como servicio (SaaS, por sus siglas en inglés), en esta categoría podemos mencionar aplicaciones ampliamente reconocidas, tales como: el correo electrónico (Gmail) y los paquetes de ofimática (Office 365). La plataforma como servicio ofrece servicios, tales como: Google App Engine, Google Borg y Docker. Estos últimos conectan los sistemas IaaS y SaaS, su enfoque es acelerar el desarrollo de aplicaciones de software [5].

El entorno PaaS tiene afinidad con la filosofía moderna para el desarrollo de software DevOPS (Developer Operators, por sus siglas en inglés), es un entorno diseñado para facilitar el desarrollo de aplicaciones y simplificar sus operaciones en un modelo inicialmente enfocado para la nube. Es decir, busca abstraer la complejidad y retos que tiene que enfrentar el ingeniero de tecnologías de la información en la configuración y gestión de sistemas rígidos de nube para que el enfoque pueda centrarse en acelerar los desarrollos de nuevas aplicaciones por parte de los usuarios de los múltiples *frameworks* y componentes tecnológicos existentes. En la Figura 1.3, tomada y modificada de [5], sé ilustra cómo el modelo de PaaS sirve como barrera para separar las tecnologías y su complejidad de configuración del usuario final, quien cada vez requiere acelerar su desarrollo haciendo uso inmediato de una pila de tecnológicas altamente disponibles [5].

En la Figura 1.4, tomada de [5], ilustra una arquitectura de PaaS. En primer lugar, las aplicaciones alojadas en la nube debe tener funciones generales de cómputo, infraestructura, entorno de desarrollo y servicios de redes informáticas. Cuando es necesaria la codificación, el PaaS generalmente ofrece opciones para el desarrollo de aplicaciones de software; construcción, relacionados con la funcionalidad de la infraestructura y su monitoreo; prueba, es la parte fundamental del hardware relacionada con los componentes

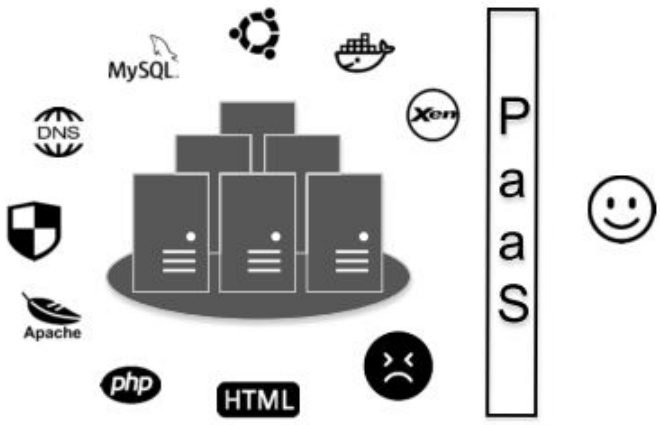


Figura 1.3: PaaS como mecanismo de abstracción tecnológica.

y servicios tales como almacenamiento, sistema operativo, red informática; la última parte está relacionada con el administrador de clúster que se encarga proveer el monitoreo de la infraestructura, gestión de recursos, programación de tareas, coordinación de concurrencia, análisis de registro del sistema y visualización de información.

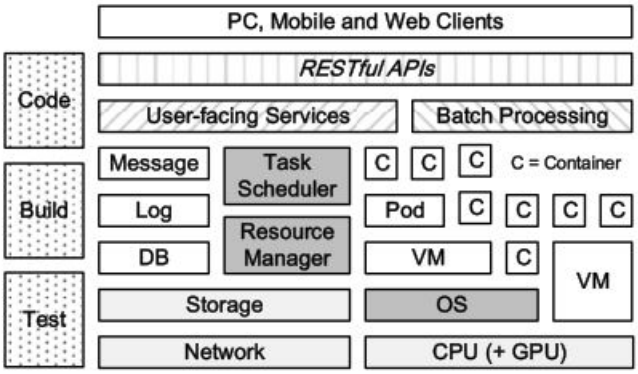


Figura 1.4: Estructura de PaaS.

Por lo tanto, los sistemas *Big Data* con la introducción de PaaS, se han visto beneficiados, pues es una forma de simplificar su uso e implementación, acelerando el desarrollo de aplicaciones relacionadas con el análisis masivo de datos. Además, la mayor parte

de los *frameworks* que dan soporte para dicha actividad se pueden implementar bajo este modelo de PaaS.

En resumen la revolución de *Big Data* y el interés de las industrias en cambiar su estrategia de inversión e integrarse al proceso de transformación digital, ponen en primer plano el conjunto de componentes tecnológicos que brindan el servicio de *Big Data*. Para los organismos relacionados al sector tecnológico, tal como El LANTI.

1.2. Descripción del Problema

El LANTI, sede UACJ, ofrece servicios de cómputo dentro y fuera de la institución. De tal manera que, las prestaciones relacionadas con el almacenamiento y cómputo de alto desempeño son parte fundamental de la existencia de esta entidad. Para un cliente o solicitante interesado en la disposición de recursos para implementación y despliegue de sistemas, tales como los que exige una plataforma *Big Data* con todos los servicios, configuraciones y elementos que permitan el uso y despliegue del mismo, pueden existir múltiples desafíos técnicos cuando su área de conocimiento no está relacionada directamente con la implementación e integración de estas tecnologías y su gestión. Por otro lado, otorgar la libertad para que cada cliente instale y ejecute sus herramientas por separado, lleva consigo un bajo aprovechamiento de estos recursos y servicios cuando existe una baja tasa de concurrencia a los mismos.

Objetivo General

Diseñar e implementar una plataforma *Big Data* con los componentes para el procesamiento, almacenamiento, análisis inteligente y visualización de datos con enfoque en la detección de anomalías.

Objetivos Específicos

1. Evaluar sistemas a nivel de clúster que soporten *frameworks* de tipo *Big Data* comunicados por una interfaz común. Derivado de dicha evaluación, implementar un clúster de cómputo.
2. Implementar y desplegar la plataforma como servicio en el clúster con los componentes *Big Data*.
3. Validar el adecuado funcionamiento de los componentes *Big Data* en la plataforma como servicio bajo la perspectiva de un sistema personalizado que puede ser gestionado bajo una interfaz común. Para este objetivo se utilizan casos de prueba enfocados en la detección de anomalías.
4. Evaluar el uso de un sistema *Big Data* bajo el esquema de PaaS personalizado en comparación a una implementación *standalone*[2] solicitada por un usuario de LANTI.

1.3. Alcances y Limitaciones

A continuación mencionamos los alcances de la implementación del PaaS para *Big Data*, tales como:

- El PaaS implementado se caracteriza por la utilización del paradigma de cómputo distribuido, compatibilidad con arquitecturas de software previamente preconfiguradas y contenerizadas, escalabilidad horizontal y vertical, prestación de servicios adicionales como aislamiento de redes y aislamiento de recursos, entre otros.
- Es tolerante a fallas, cuando un servidor se daña o queda fuera de funcionamiento, de manera automática se reorganizan las cargas de trabajo y replicación de datos.

[2] Según https://bit.ly/3yZyKqx es una implementación con sus propios recursos y generalmente no interactúa con otros servicios o aplicaciones

- Puede instalarse en sitio de manera local o a través de los servicios de un proveedor de nube pública, tales como: Amazon, Azure y Google Cloud.

Entre las limitaciones encontradas se encuentra las siguientes:

- Los recursos e infraestructura computacional requerida deben estimarse previamente si no cuenta con una infraestructura elástica de nube.
- El manejo y operación de los servicios *Big Data* puede ser una dificultad inicial de la plataforma si no se está familiarizado previamente con el uso de esta clase de prestaciones.
- Algunos servicios proporcionados por las herramientas de software del clúster son accesibles únicamente por la interfaz de programación de aplicaciones (API, por sus siglas en inglés) o interfaces de desarrollo interconectadas.

1.4. Justificación

El cómputo en la nube aborda soluciones para todas las áreas de la ciencia y la tecnología computacional. Sin embargo, los ingenieros expertos en todos los componentes incorporados a la nube son escasos. Según [5], el nivel de experiencia de estos profesionales generalmente se ve limitado al dominio de un solo nivel de servicio. Por ejemplo, el uso y gestión de IaaS, configuración de servicios web, administración y configuración de SaaS se administran por más de un especialista.

Como resultado, los ingenieros de computación en la nube con un déficit en todas las áreas deben invertir un tiempo mayor al tener que lidiar con la complejidad de configuraciones relacionadas con el hardware, sistema operativo y administración de servicios, tales como: contenedores, base de datos, red, sistema de nombres de dominio y el cortafuegos, por nombrar algunos [5].

Citando una de las principales actividades de El LANTI, según aparece en [13], se resume como:

«La prestación de recursos de cómputo de alto rendimiento y el impulso del desarrollo tecnológico-científico»

En ese sentido, el modo de implementación de una plataforma como servicio de *Big Data*, contribuye en la integración de un un nuevo paradigma en el uso y gestión de recursos en LANTI al desplegar arquitecturas contenerizadas bajo un esquema de cómputo distribuido y aislamiento de recursos. Entre los impactos del proyecto se mencionan los siguientes:

- **Impacto Social:** Este tipo de implementación puede impulsar el desarrollo de proyectos científico-tecnológicos para el desarrollo de aplicaciones donde se involucra el análisis masivo de datos.
- **Impacto Académico:** La implementación de este proyecto puede contribuir al aprendizaje y fortalecimiento de temas a través de la experimentación directa con el PaaS de temas, tales como: contenedores, cómputo distribuido, manejo y administración de servicios y otros elementos relacionados con tecnologías emergente de nube.
- **Impacto Tecnológico:** La implementación, adicionalmente, tiene prestaciones, tales como: capacidad de integrar componentes de software que permiten la programación de aplicaciones relacionadas con el desarrollo de software y microservicios, Internet de las Cosas, *Big Data*, tecnologías de bases de datos NoSQL (Lenguaje No Estructurado de Consultas) y gestión de contenedores.
- **Impacto Económico:** Este proyecto puede contribuir a la competitividad de las organizaciones facilitando la implementación de el entorno de desarrollo *Big Data* a través de la generacion de aplicaciones que ofrezcan nuevas y diversas gamas de servicios digitales, tales como: aplicaciones relacionadas con monitoreo de dispositivos en tiempo

real y sistemas dirigidos por eventos, analítica predictiva para detectar anomalías en procesos industriales, por mencionar algunas.

1.5. Metodología

La metodología utilizada es considerada clásica en cualquier rama de ingeniería, surgió para dar soporte a los procesos de diseño desde que nace la idea de una solución hasta su retiro o reemplazo, se originó en el siglo pasado y aunque es muy similar a la metodología cascada para desarrollo de software, esta es aplicable no solo al desarrollo de software [14].

Esta metodología consiste en seguir un proceso secuencial y general de resolución de problemas que se puede aplicar en distintas ramas ingenieriles utilizando un diseño claro y pertinente durante todas las fases del desarrollo un proyecto. Entre sus atributos encontramos que es diseñado por y para personas, tiene un objetivo específico que cumplir para contribuir a la consecución de las propuestas de valor de la parte interesada en un contexto más amplio [14], [15].

La Figura 1.5 tomada de [15], ilustra el ciclo de vida de un producto. Primero, el sistema de interés se estudia ampliamente en su fase de definición para que posteriormente se pueda construir o implementar, probar y poner en producción. Finalmente, el soporte, mantenimiento y retiro del producto en funcionamiento dependerá del ciclo de vida o interés de la organización por establecer nuevas y mejores tecnologías que puedan surgir a futuro.

Figura 1.5: Metodología para el ciclo de vida de un producto.

La metodología utilizada en el proceso de implementación es compatible con las distintas disciplinas de la ingeniería y ciencias [15]. Aborda los siguientes pasos:

Análisis y diseño

En esta fase se investigan las plataformas que cumplen con los requerimientos de compatibilidad de *frameworks* y componentes actualizados de *Big Data* que son administrables bajo una interfaz común en un modelo PaaS.

Construcción

En esta fase inicia el proceso de implementación del clúster y habilitación de la plataforma como servicio con la instalación de los componentes necesarios para realizar los procesos de mineria y analítica avanzada de datos.

Pruebas

En esta fase se realizan pruebas del PaaS, componentes de software y entornos de desarrollo utilizando casos de prueba.

Operación, mantenimiento, desuso o reemplazo

Esta fase queda limitada al monitoreo interno y actualizaciones que permitan mantener un adecuado funcionamiento de la plataforma implementada en El LANTI.

Capítulo 2

Marco Referencial

Este capítulo se divide en Marco Teórico Conceptual y Marco Tecnológico. En el primero, se describen las tecnologías generales relacionadas en un entorno de plataforma como servicio, fundamentos teóricos de *Big Data* y temas de aprendizaje computacional. La segunda, aborda el conjunto de tecnologías disponibles en el mercado para procesamiento, almacenamiento y análisis avanzado de datos.

2.1. Marco Teórico Conceptual

En esta sección son presentados los conceptos, definiciones y teorías generales para ofrecer una mejor comprensión de los elementos que constituyen el entorno de un PaaS, tales como: la cultura DevOps, donde es importante el enfoque de desarrollo directamente observando la operación o proceso que será implementado en un sistema de software. Antecedentes de cómputo de alto rendimiento, que es crucial en los PaaS donde la cantidad de recursos de cómputo es elevada y requiere una gestión eficiente de los mismos. Teorías generales de *Big Data* y su implicación en la detección de anomalías, así como, los algoritmos de aprendizaje computacional que son el soporte para la búsqueda de estas últimas.

2.1.1. Componentes generales de una plataforma como servicio

A continuación se abordan conceptos y teorías directamente relacionados al PaaS de la implementación presentada en este reporte de investigación.

Cultura DevOps

Según [16], la cultura DevOps permite alinear las operaciones con las necesidades comerciales y del cliente. En un entorno DevOps, tanto los desarrolladores y administradores de sistemas comparten la responsabilidad para cumplir con los requisitos de una solución a tiempo. Por consiguiente, ambos comparten las tareas con el mismo grado de importancia. Para el caso del desarrollador su tarea principal es asegurarse de que el software pueda cumplir con las altas requisitos de disponibilidad. Por otro lado, personal de estrategias y operaciones trabaja en estrecha colaboración con los desarrolladores para proporcionar información sobre la implementación y el proceso de desarrollo todo conformado por un solo equipo.

El ciclo de desarrollo en DevOps [16], debe ser un conjunto uniforme de procedimientos y procesos que den como resultado un producto terminado: el servicio. No existe el concepto de "ellos", como en "entregar a ellos"; sólo existe "nosotros", el equipo trabajando en el producto. Los miembros del equipo son en gran parte personas que trabajan con una visión general, sin embargo, no dejan de ser especialistas de un área en específico. La mayoría de las organizaciones de DevOps se centran en objetivos comerciales claros, como escala, eficiencia y alto tiempo de actividad. Haciendo hincapié en las personas y los procesos en lugar de soluciones a la medida.

Antecedentes del cómputo de alto rendimiento

Según [17], este tipo de computación esta orientado para soportar aplicaciones que demandan un alto desempeño y coordinación eficiente de recursos computacionales durante su ejecución. Ámbitos científicos, ingenieriles y militares, han utilizado el este tipo

de cómputo debido a sus altas exigencias en el desempeño de sus sistemas informáticos.

El objetivo del paralelismo en un principio fue agilizar el tiempo de ejecución de algoritmos de programación. Las bases teóricas de este tipo de cómputo fueron establecidas en los años setenta y los primeros equipos comerciales con estas capacidades surgieron a mediados de los años ochenta. En consecuencia, el acelerado crecimiento del procesamiento paralelo fue motivado por las exigencias del mercado en ese momento. El alto desempeño, inicialmente abordaba arquitecturas y tecnologías optimizadas para los requerimientos que demandaban las aplicaciones de alta disponibilidad en aquella época. Sin embargo, el costo de estos equipos de cómputo no era accesible para cualquier usuario, principalmente por el declive económico de la época de fines de los ochenta [17].

El desarrollo acelerado de nuevos microprocesadores, cada vez más potentes y económicos, facilitó el acceso al cómputo de alto desempeño a grupos de personas y organizaciones. De igual manera, el modelo de cómputo en la nube actualmente permite el uso de recursos bajo demanda, lo que significa que cualquier usuario puede acceder a una cantidad ilimitada de hardware y utilizar el paralelismo por un costo accesible.

Aplicaciones de la computación paralela

El cómputo de alto rendimiento permitió el desarrollo de aplicaciones para procesar complejos modelos de programación que contribuyeron a comprender fenómenos naturales en ciencias geológicas y climáticas, tales como pronóstico del clima a corto plazo, así como la polución, por mencionar algunos.

Además, el desarrollo de nuevos modelos matemáticos, como el de descomposición de dominios permitió desarrollar nuevas aplicaciones científicas que impulsaron el desarrollo en ciencias químicas, genéticas y nucleares gracias a los resultados de cálculos intensivos usando computación paralela. El diseño asistido por computadora también se benefició de este modelo de cómputo a través de la ejecución de complejos cálcu-

los. Por consiguiente, los procesos en el diseño de vehículos y aeronaves a través de la simulación se vieron mejorados [17].

Fundamentos teóricos

El procesamiento paralelo y distribuido difiere de la computación secuencial clásica por la gran diversidad de soportes existentes para cada una de ellas. Los sistemas paralelos están enfocados hacia procesadores potentes que están organizados por una red jerárquica de interconexión de múltiples etapas y un alto volumen de memoria local. Por otro lado, la aparición de la computación en clúster y la metacomputación que según [18], se refiere a una red de recursos computacionales heterogéneos vinculados por software utilizan algunos modelos para abstraer las características principales de los componentes físicos y los mecanismos lógicos [17].

Es decir, el hardware deja de ser un obstáculo principal en la ejecución del cómputo paralelo. Según se menciona en [19], el paralelismo se encuentra en todos los niveles de las arquitectura de un equipo de cómputo moderno, tales como:

- Arquitectura de microprocesadores. El objetivo principal se ha convertido en el diseño de un procesador capaz de ejecutar varias instrucciones simultáneamente.
- Múltiples núcleos. Cada uno de los cuales es capaz de ejecutar su propio flujo de instrucciones. Si estos últimos están diseñados para que los núcleos colaboren en la ejecución de una aplicación, se dice que esta se ejecuta en paralelo y puede acelerarse considerablemente su desempeño.
- Varios procesadores multinúcleo. Un servidor con estos componentes es capaz de ejecutar uno o varios servicios en paralelo.
- Procesadores gráficos. Son capaces de ejecutar cientos o incluso miles de hilos en paralelo. Un ejemplo de su aplicación es la animación gráfica.

Definición de sistema distribuido

Consiste en un conjunto de equipos de cómputo que están interconectados usando redes informáticas de alta velocidad, cada uno con su propia unidad de procesamiento y su memoria física. Para dichos sistemas, el avance en la tecnología de redes de comunicación relacionados con el creciente aumento en la velocidad y ancho de banda ha sido beneficioso por las mejoras en la tecnología de transmisión (procesamiento de señales y transmisión óptica e inalámbrica), impulsando el desarrollo de la informática distribuida. Las computadoras interconectadas son ahora sistemas que pueden ser confiables, rápidos, escalables y baratos [17].

El cómputo de alto rendimiento es uno de los elementos clave para poder procesar y analizar datos, especialmente en cantidades masivas. Sin embargo, otra opción muy extendida en la actualidad es el cómputo en la nube, concepto que se presenta en la siguiente subsección.

Cómputo en la nube

Es un modelo de cómputo ubicuo, ofrece un conjunto de recursos informáticos bajo demanda [20], tales como: redes, servidores, almacenamiento, aplicaciones y servicios a través de un entorno dedicado por Internet. Los elementos mencionados se pueden utilizar y acceder desde cualquier lugar gracias al uso de interfaces web y una interacción mínima con el proveedor de servicios. Este modelo de nube se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación según [20]:.

Características:

- **Autoservicio**. Esta característica permite que un usuario de nube pueda acceder a los recursos de infraestructura sin depender de un intermediario y en todo momento. Por ejemplo, almacenamiento, bases de datos, aplicaciones, instancias de máquinas virtuales. De ese modo, salvo en casos especiales puede solicitar soporte técnico del proveedor de nube.

- **Acceso por red**. Las capacidades de la infraestructura de hardware y software están disponibles a través de la red de internet. El acceso se lleva a cabo utilizando mecanismos estándar, tales como: navegadores web y plataformas heterogéneas de clientes ligeros o de manera estándar (por ejemplo, teléfonos móviles, tabletas, computadoras portátiles y estaciones de trabajo).
- **Acceso común de recursos**. Los recursos informáticos del proveedor de nube se agrupan para servir a múltiples usuarios utilizando un modelo conocido como *Multitenant* con diferentes recursos físicos y virtuales repartidos dinámicamente de acuerdo con la demanda del solicitante. Existe una sensación de independencia de ubicación, en el sentido que el cliente generalmente no tiene control o conocimiento sobre la ubicación exacta de los recursos proporcionados, pero puede especificar la ubicación en un nivel más alto de abstracción (por ejemplo, país, estado o centro de datos). Entre la clase de recursos que operan bajo este modelo incluyen almacenamiento, procesamiento, memoria y ancho de banda de red.
- **Elasticidad**. Las capacidades de cómputo se pueden aumentar elásticamente sin restricción y pueden ser accedidas en cualquier momento. Es decir, si los recursos de hardware han sido agotados o llegan a su límite de acuerdo con la demanda del cliente de nube.
- **Servicio medido**. Los sistemas en la nube controlan y optimizan automáticamente el uso de los recursos al aprovechar una capacidad de medición en algún nivel de abstracción apropiado para el tipo de servicio (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas). El uso de los recursos se puede monitorear, controlar e informar, proporcionando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

Modelos de servicio

Los modelos de servicio, que se conocen actualmente y que se pueden decir que se han

establecido como oficiales [20] son los siguientes:

- **Software como servicio SaaS**. Se define como el uso de las aplicaciones del proveedor de nube. Ejecutadas en su propia infraestructura y accedidas por dispositivos cliente a través de una interfaz como un navegador web o una interfaz de programa. El consumidor no gestiona ni controla la infraestructura de nube subyacente, incluida la red, los servidores, los sistemas operativos, el almacenamiento o incluso las capacidades de las aplicaciones individuales.
- **Plataforma como servicio PaaS**. Se refiere a la implementación de aplicaciones mediante lenguajes de programación, bibliotecas, servicios y herramientas compatibles con el proveedor de nube. El usuario de nube no gestiona la infraestructura sobre la cual se ejecuta la aplicación, como: red, servidores, sistemas operativos o el almacenamiento. Sin embargo, si controla las aplicaciones desarrolladas, las configuraciones de la aplicación y entorno de hospedaje.
- **Infraestructura como servicio IaaS**. Es la capacidad que se brinda al usuario de nube para la adquisición de recursos computacionales, tales como: procesamiento, almacenamiento, redes y otros recursos informáticos fundamentales. Sobre esta capa de recursos el usuario puede implementar y ejecutar software arbitrario, puede incluir sistemas operativos y aplicaciones. En algunos casos puede controlar componentes de red seleccionados, por ejemplo, firewalls de host.

Modelos de implementación

Los modelos de implementación o despliegue son los siguientes según [20]:

- **Nube privada**. La infraestructura de la nube está adquirida para uso exclusivo de una sola organización o empresa. Comprende múltiples usuarios (por ejemplo, unidades de negocios). Puede ser propietaria, gestionada por la organización, un tercero o alguna combinación de ellos, y puede existir dentro o fuera de las

instalaciones, ver Figura 2.1 donde se ilustran los tipos de nube[1] .

- **Nube comunitaria**. La infraestructura de la nube se adquiere para uso exclusivo de una comunidad específica de usuarios de organizaciones que tienen preocupaciones compartidas, por ejemplo: misión, requisitos de seguridad, políticas y consideraciones de cumplimiento. Puede ser utilizada y gestionada por una o más de las organizaciones de la comunidad, un tercero o alguna combinación de ellas, y puede existir dentro o fuera de las instalaciones
- **Nube pública**. La infraestructura de la nube está disponible para uso abierto por el público en general. Puede ser propiedad, administrada y operada por una organización empresarial, académica o gubernamental, o alguna combinación de ellas. Existe en las instalaciones del proveedor de la nube.
- **Nube Híbrida**. Es una composición de dos o más infraestructuras de nube distintas. Privadas, comunitarias o públicas que siguen siendo entidades únicas. Además, están unidas por una tecnología estandarizada o patentada que permite la portabilidad de datos y aplicaciones.

Tecnología de contenedores Docker

Es una tecnología de software que encapsula aplicaciones y los elementos necesarios para su funcionamiento (dependencias) en unidades individuales nombradas contenedores [21]. Sin embargo, también se conocen como máquinas virtuales ligeras, debido a que cuentan con un sistema de archivos independiente y virtualizado, procesos que se ejecutan de manera aislada y comparten el núcleo de un sistema operativo (SO) anfitrión del contenedor. El clúster DCOS implementado en este trabajo utiliza y gestiona todos sus paquetes de software con el esquema de contenedores, por eso es importante conocer el concepto antes mencionado.

[1] Figura tomada de `https://bit.ly/3pQ9wad`

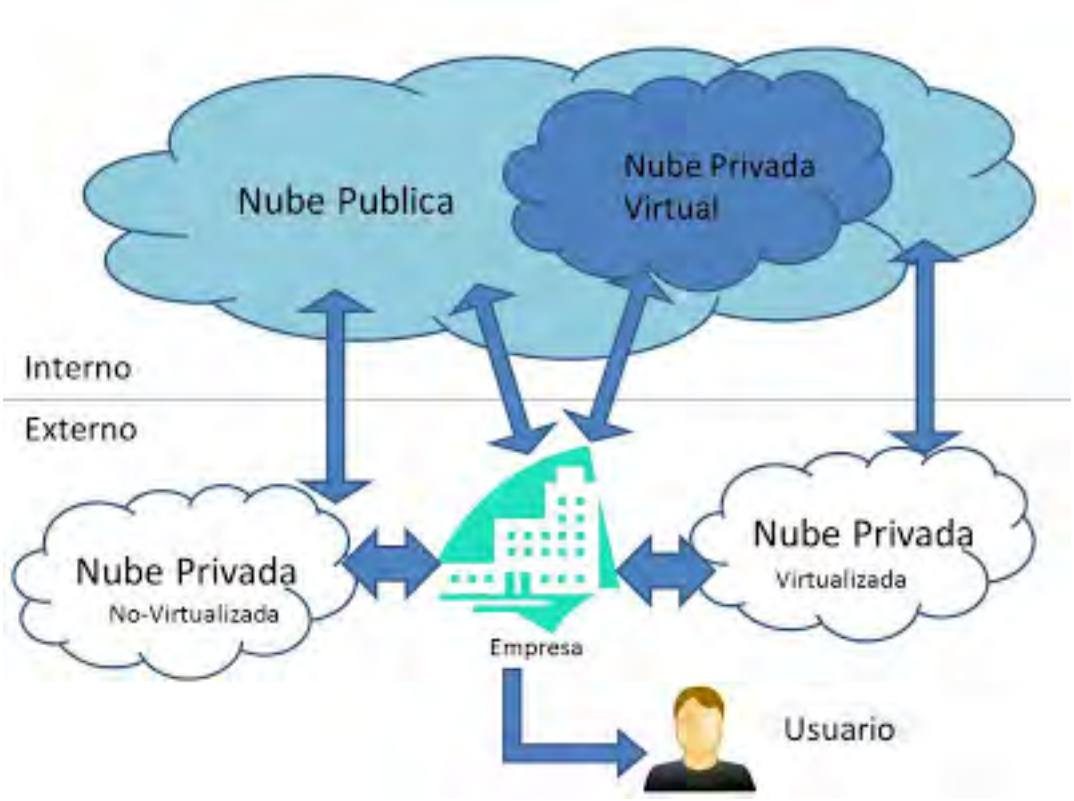


Figura 2.1: Tipos de implementación.

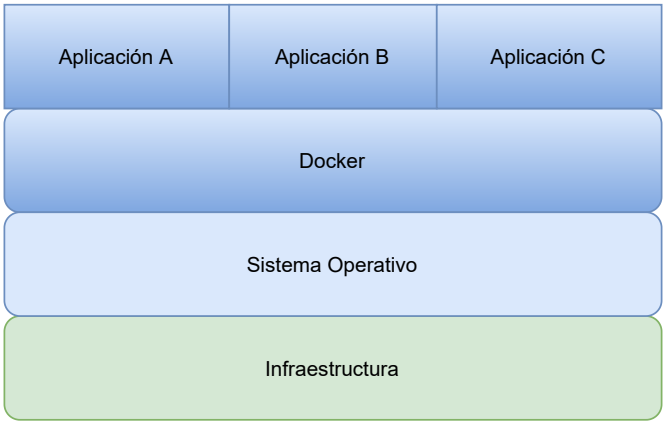


Figura 2.2: Arquitectura Docker.

El cómputo de alto rendimiento combinado con la tecnología de contenedores por medio de una infraestructura local o de nube, es el primer elemento necesario para la ejecución de componentes *Big Data*. Sin embargo, también son requeridos algoritmos de *Machine Learning*, estos permiten el análisis de los datos para encontrar nuevos significados, patrones y elementos que otorguen un nuevo valor a la información.

2.1.2. Fundamentos teóricos de *Big Data*

En esta sección se presentan las teorías relacionados a *Big Data* para una mejor comprensión de las capacidades que puede desplegar la plataforma como servicio a través de sus componentes tecnológicos interconectados.

Definiciones

Se refiere a una serie de tareas enfocadas a la adquisición y procesamiento de un amplio y complejo volumen de datos, normalmente agrupados en conjuntos [22], [23], [24]. Sin embargo, la definición original fue acuñada por el modelo de Laney [7], el cual define el término como: datos que poseen los atributos de las tres 'V'(Variedad, Volumen, Velocidad).

En ese sentido, un concepto que amplía el original es el de 5'V'(Variedad, Veracidad, Valor, Volumen, Velocidad), este sugiere dos nuevas características para la información, tales como: Veracidad y Valor [25]. En la Figura 2.3 tomada de [25], se ilustran las propiedades mencionadas. Volumen se refiere a la cantidad de los datos; velocidad se refiere a la ágil y rápida tasa de transferencia con la que deberían ser trasladados los datos de una fuente a otra; variedad es un indicador de tipos de datos y fuentes heterogéneas; la veracidad describe la consistencia e integridad de los datos; valor son nuevos significados encontrados en la información y traducidos a conocimiento. Por lo tanto, este hecho mejora el soporte en la toma de decisiones, lo que contribuye a tener

un impacto positivo en la organización. Aspectos tales como: económicos, calidad de los productos, mejora de servicio, son algunos de los beneficios obtenidos con la cualidad de valor [26].

Según [27], la digitalización y la inclusión de la tecnología en aspectos cotidianos, tales como: el hogar, trabajo, salud y entretenimiento ha fomentado la utilización de dispositivos inteligentes. Es decir, teléfonos modernos, computadoras, tabletas, relojes inteligentes y el uso de plataformas de redes sociales, servicios de *streaming* (para audio y vídeo), por mencionar algunos, contribuyen a la generación de datos masivos. Por consiguiente, el aumento exponencial en la información, representa una oportunidad para extraer valor y encontrar nuevos patrones en la información que pueden ser explotados al implementar nuevos servicios, productos, campañas de mercadeo, ahorro de costos, produciendo un beneficio en una organización, empresa o negocio. Según [28], algunas características inherentes de *Big Data* son:

Figura 2.3: Resumen de las 5V.

- Integración de datos estructurados y no estructurados.
- Énfasis en la velocidad, escalabilidad, movilidad, seguridad, flexibilidad y estabilidad.
- El tiempo de generación de la información es fundamental para extraer valor de varias fuentes de datos, incluyendo dispositivos móviles, identificación por radiofrecuencia y tecnologías de sensores automatizadas.

En la actualidad, las sociedades más desarrolladas tecnológicamente generan grandes volúmenes de datos en periodos relativamente cortos de tiempo. Cabe agregar que las tecnologías para procesar esos datos también están evolucionando y algunas de las cuales se presentan en este Capítulo 2 en las secciones Marco Referencial y Tecnológico.

En la siguiente subsección abordaremos las aplicaciones más importantes en las que *Big Data* puede ser aplicado tanto en industria como en la sociedad.

Aplicaciones en industria y sociedad

Según [7], la cantidad de datos generados colectivamente, por organizaciones e individuos, crece exponencialmente. En décadas anteriores, las capacidades de cómputo y almacenamiento no permitieron lograr este volumen de información en un corto periodo de tiempo pues representaba un desafio técnico y de elevados costos. Actualmente, es posible gestionar cantidades tan grandes como el exabyte, equivalente a un mil millones de gigabytes, administrables gracias al uso e integración de nuevos componentes tecnológicos.

Big Data, puede contribuir a encontrar un valor en la información para mejorar la competitividad de una organización [8]. PromptCloud [29], reportó en 2016 un crecimiento de en las industrias de 25.2 billones de dolares, USD, debido al uso de *Big Data* y analíticas avanzadas sobre los datos que permitieron la generación de nuevas estrategias de negocio. Tres tendencias han revolucionado *Big Data*, según [8], son las siguientes:

- Un aumento exponencial de los datos disponibles según reporte de [30] y [8]. En la Figura 2.4, tomada y modificada de [30], muestra un incremento exponencial de los datos anualmente en el periodo 2010 a 2024.
- Incremento del poder computacional y la capacidad de almacenamiento, además de la reducción de los costos de estos últimos.
- Avances en los métodos de *Machine Learning* como herramientas disponibles para el análisis de complejos conjuntos de datos.

Según [9], *Big Data* está presente fundamentalmente en tres dominios de aplicación, éstos son: sociedad, productos-dispositivos y red operacional-telecomunicaciones, en cada uno de ellos se puede presentar un contexto de la detección de anomalías, por ejemplo: en manufactura, un proceso de manufactura con errores de difícil detección; en finanzas, comportamientos inadecuados con motivo de fraude por usuarios de un banco; en red operacional, denegación de servicios en una red informática a causa de ciberataques. En la Figura 2.5, tomada y modificada de [9], se ilustran de manera general dichos dominios de aplicación y se describen de la siguiente manera:

1. Sociedad. Este dominio incluye el estudio de comportamientos y patrones de consumo de las personas incorporadas en una sociedad delimitada por una geografía particular. El objetivo es diseñar campañas personalizadas de publicidad, medir el impacto de estrategias de mercadeo y fidelizar al consumidor, en el caso de detección de anomalías, detectar porque los grupos compran un determinado producto en lugar de otro del mismo precio. En esta misma categoría se encuentran también los empleados, de estos últimos se espera conocer su efectividad, valor en la empresa y satisfacción al colaborar en la misma.
2. Productos, servicios y dispositivos. Este dominio aborda los productos comerciales, servicios y dispositivos. Para detección de anomalías, se pueden presentar casos en los que a través de *Big Data* y algoritmos de analítica de datos se puede detectar la

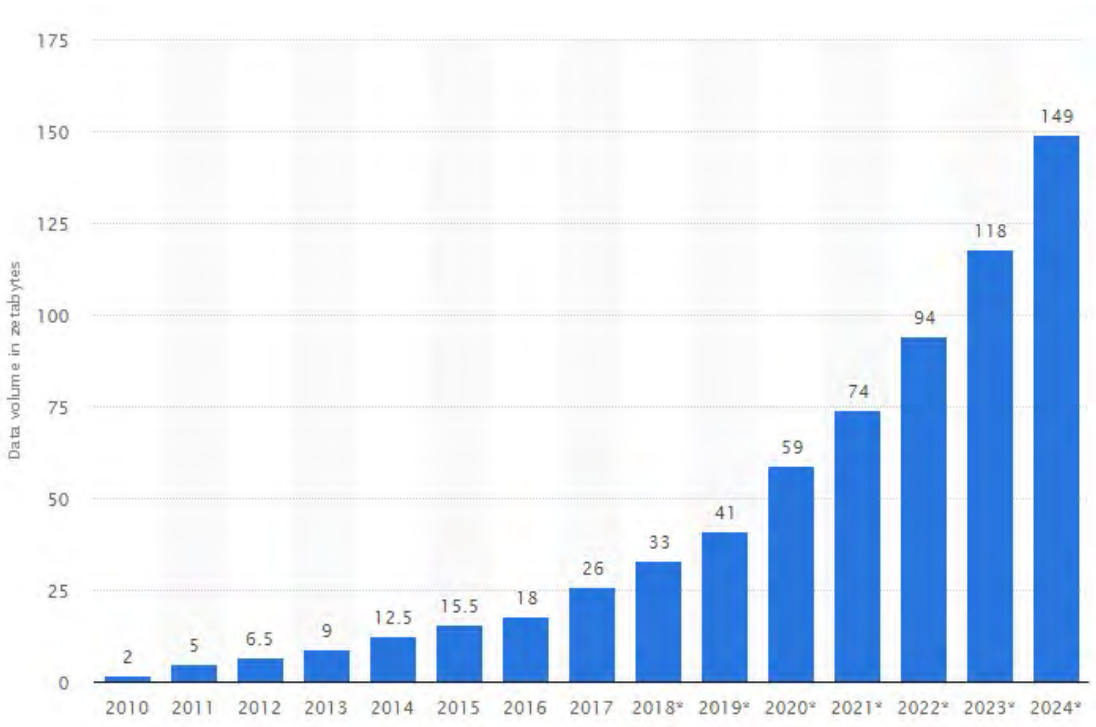


Figura 2.4: Aumento exponencial de los datos.

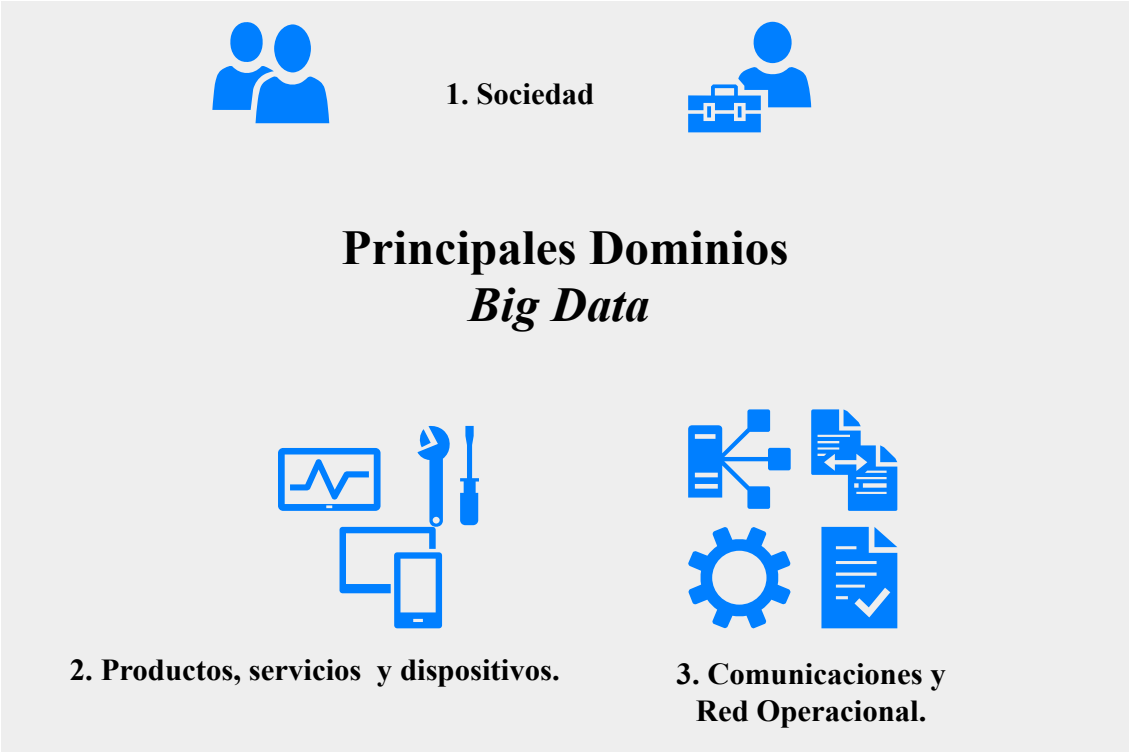


Figura 2.5: Dominios principales Big Data.

anomalía en el servicio que está propiciando que los clientes abandonan y cancelan el servicio. Por ejemplo, si la empresa vende conectividad de internet y aparecen constantes fallas e intermitencia en el servicio, el consumidor puede optar adquirir un servicio nuevo con otro proveedor. Para dispositivos se pueden registrar los datos de uso y mantenimiento que sean factibles de analizar y detectar anomalías por falta de mantenimiento.

3. Comunicaciones y Red operacional. Este dominio aborda predicción de la demanda que tendrá que suplir un negocio sobre un producto o servicio, soporte a la capacidad en la planeación de proyectos, redes de telecomunicaciones, por ejemplo: optimización del tráfico de red, seguridad, capacidad y reducción de los tiempos fuera de servicio, entre otros. En detección de anomalías existen casos particulares, es decir, la cadena de suministro se ve interrumpida por una anomalía climática que impide el reparto de materia prima a una empresa solicitante, otro ejemplo, el servicio de red funciona intermitentemente por un ciberataque no detectado en las primeras horas que ocurre dicha intrusión.

2.1.3. Fundamentos teóricos del *Big Data Analytics*

La extracción de valor de los datos (*Data Analytics*, por sus significado en inglés) se realiza a través de la explotación de modelos de aprendizaje automático que sirven como soporte en toma de decisiones a organizaciones que buscan reducción de costos, mejoras en procesos y reducción de desperdicios [31].

En esta sección se abordan los conceptos generales del aprendizaje computacional y conceptos derivados de este campo de la IA. En un contexto práctico, estos conceptos son puestos a prueba en el modelo PaaS que se implementó en este trabajo y que se describe en el Capítulo 3.

Aprendizaje computacional

En el siglo pasado, el término *Machine Learning* se dio a conocer por Arthur Samuel [32], que define aprendizaje computacional como: "Campo de estudio que otorga a las computadoras la capacidad de aprender sin una programación explicita", otra definición clásica, según [33], es un programa de computadora que aprende de la experiencia E respecto a una tarea T y desempeño medido por P, si dicho desempeño sobre T, es medido por P, mejora con la experiencia de E. Según [34], es una rama de la Inteligencia Artificial que tiene por objetivo automatizar el aprendizaje; además, está relacionado con diversos campos de estudio, tales como: estadística, filosofía, teoría de la información, biología y la complejidad computacional.

Machine Learning está ligado ampliamente con el reconocimiento de patrones. Sin embargo, este tiene su origen en la ingeniería, mientras que el aprendizaje automático creció fuera de la informática [35]. Por consiguiente, este último es impulsado por el reconocimiento de patrones y ambos campos de estudio han experimentado un desarrollo importante en las últimas décadas. Los casos de prueba para detección de anomalías presentados en el Capítulo 4, utilizan algoritmos de aprendizaje automático, por tal motivo se presentan las definiciones básicas, clasificación y aplicaciones generales que puedan facilitar al lector la comprensión de dichos casos con un enfoque objetivo.

Preeliminares del aprendizaje computacional

A continuación se describen algunos elementos conceptuales que aparecen de forma constante en el tema de *Machine Learning*.

Algoritmo. Según [36], es un procedimiento computacional bien definido que toma un valor o conjunto de valores como entrada y produce una salida con uno o varios resultados. También podemos ver un algoritmo como una herramienta para resolver un problema computacional bien especificado. La declaración del problema especifica en términos generales la relación entrada-salida deseada. El algoritmo describe un procedimiento

computacional para lograr esa relación entrada-salida.

Representación. Según [37], este concepto se ocupa de la representación del conocimiento a través de conjuntos de reglas, incluidos árboles de decisión, máquinas de vectores de soporte, instancias, redes neuronales, modelos gráficos, conjuntos de modelos.

Modelo. Según [38], Es una metodología que se aplica a los datos para la identificación de patrones.

Elementos del conjunto de datos

A continuación se define Según [39], conjunto de datos y sus particiones, tal como:

- Conjunto de datos (*dataset*). Es el total de los datos organizados de forma tabular. Como se puede ver en la Figura 2.7 tomada y modificada de [39], los atributos son representados por columnas etiquetadas desde x_1,...x_3. Por ejemplo, los parámetros clínicos de un paciente pueden ser representados por estos atributos, tal como: glucosa, presión arterial, peso, edad. La columna y, es la variable objetivo que tiene el resultado de una clasificación. Siguiendo el ejemplo del paciente, y puede almacenar el resultado si un paciente padece diabetes o no.
- Conjunto de entrenamiento (*training set*). Es una parte del total del conjunto de datos (aproximadamente 60 %) para entrenar el modelo de aprendizaje automático. En la Figura 2.6, se ilustra la sección de datos correspondiente al *training set* del conjunto total.
- Conjunto de prueba (*test set*). Es un 20 % del total del conjunto de datos utilizado para evaluar correctamente el rendimiento de un algoritmo de aprendizaje automático. En la Figura 2.6, se ilustra la sección correspondiente al conjunto de prueba.
- Conjunto de validación. (*validation set*), es un 20 % del total del conjunto de datos (ver Figura 2.6). Esta parte del conjunto es utilizada para mejorar el modelo.

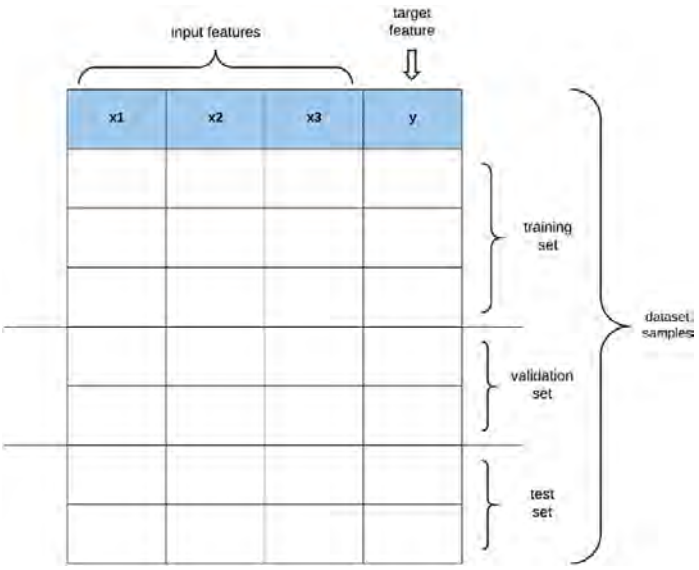


Figura 2.6: Dataset y sus particiones.

Aprendizaje supervisado

También es conocido como aprendizaje inductivo y se subdivide en clasificación y regresión. El primero, según [40], se enfoca en determinar la función $f(x)$ que recibe como entrada un vector x conocido como muestra [39], con parámetros relacionados al problema que será analizado. Un problema de clasificación es por ejemplo, detectar si un correo electrónico es genuino o sospechoso o si un tumor es maligno o canceroso [41].

Considere el el reconocimiento de dígitos escritos a mano de la Figura 2.7 tomada y modificada de [35], cada uno de los cuales corresponde a una imagen de 28×28 píxeles, por lo tanto, puede representarse mediante un vector x. El objetivo del aprendizaje supervisado es determinar una función $f(x)$ que al recibir los parámetros de entrada del vector x producirá como salida la identificación de un dígito del siguiente conjunto:

$$\{0, ..., 9\}$$

Es decir, si el vector de entrada corresponde al número cinco, la función $f(x)$ debe dar como resultado en su salida cinco. Este es un problema no trivial de reconocimiento

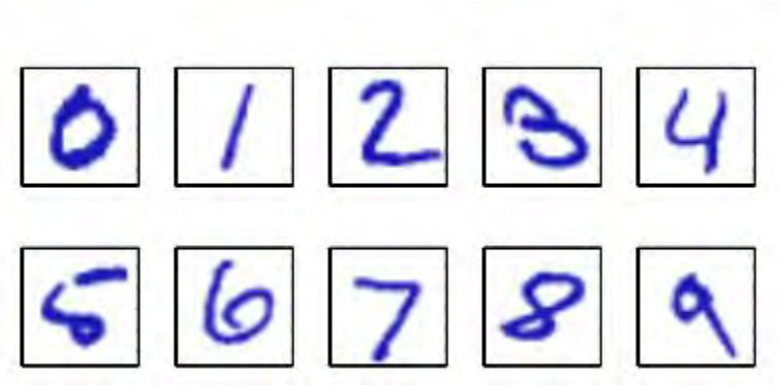


Figura 2.7: Dígitos manuscritos de los números naturales.

de patrones debido a la amplia variabilidad de la escritura a mano. Un enfoque de aprendizaje automático es considerar un conjunto de N dígitos

$$\{X_1, ..., X_N\}$$

llamado conjunto de entrenamiento para representar las categorías de los dígitos que se conocen de antemano y han sido previamente etiquetados individualmente usando un vector objetivo t, que representa la identidad del dígito correspondiente. De ese modo, para cada vector t existe una imagen de dígito x.

La regresión consiste en determinar los valores de la variable objetivo que es otra forma de nombrar a la función $f(x)$ [39] y que son expresados como números de valor real.

Por ejemplo, un conjunto de datos de vivienda , ver Figura 2.8 tomada y modificada de [39], tiene características representadas por cada columna, etiquetadas como: el precio de la casa (Precio), el número de camas (No.Camas), el número de baños (No.Baños), y pies cuadrados totales (Pies Cuadrados). En este caso, a través de un modelo predictivo se predice el precio del inmueble.

De ese modo, en el aprendizaje supervisado cada tarea debe ser apoyada por un experto del dominio de aplicación de la industria o ciencia donde surge el problema, esto con

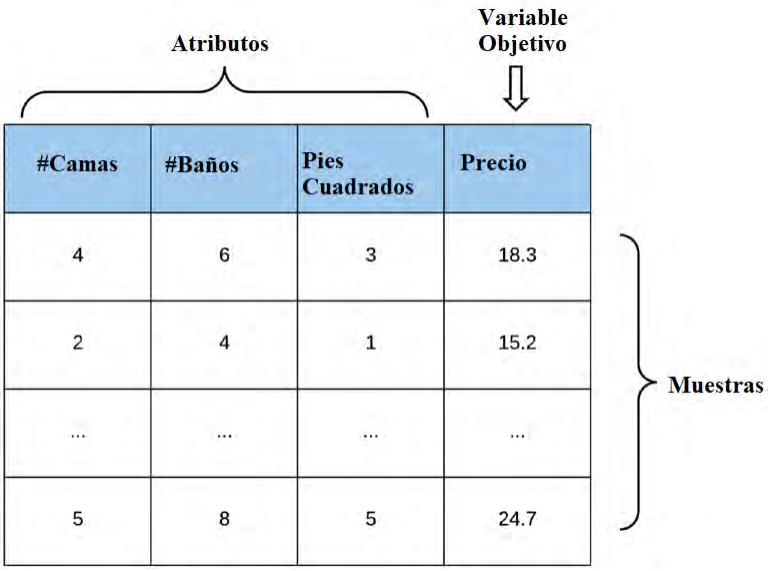


Figura 2.8: Ejemplo de un conjunto de datos.

el fin de validar las salidas de la función f [40] . Además, este proceso es apoyado por un denominado aprendiz (learner, por su significado inglés), el cual tiene por objetivo desarrollar la capacidad de generalizar y asociar.

En este aprendizaje, un algoritmo se entrena con la ayuda de un conjunto de "ejemplos" ya definidos. Dicha actividad contribuye con aprendizaje para formular un preciso utilizando los datos recién ingresados con facilidad y sin ninguna interferencia [37].

Aprendizaje no supervisado

El enfoque principal de este tipo de aprendizaje es generar conjuntos de datos con atributos similares [39]. A estos conjuntos normalmente se les denomina clústeres. En la Figura 2.9 tomada y modificada de [41], se ilustra el ejemplo de lo descrito anteriormente, datos dispersos son representados por los cuadrados de un solo color en la Figura 2.9 más a la izquierda, después de la aplicación de un algoritmo de segmentación, los datos son asociados y agrupados (generación de clústeres), ver Figura 2.9 parte derecha. Un enfoque también importante de este tipo de aprendizaje es su capacidad para determi-

nar la distribución de datos dentro de un espacio de entrada, tarea que se conoce cómo estimación de densidad. La segmentación puede proyectar datos en una espacio de dos o tres dimensiones con fines de visualización [41].

Figura 2.9: Clústering: agrupación por atributos similares.

Algoritmos de aprendizaje supervisado

En esta sección se abordan los algoritmos de aprendizaje supervisado: árboles de decisión y redes neuronales artificiales utilizados para validar casos de prueba relacionados con anomalías.

Árboles de decisión

Es uno de los clasificadores más utilizados en estadística y aprendizaje automático. El árbol de decisiones es un diseño jerárquico que implementa el enfoque de dividir un problema complejo en partes más simples. Es una técnica no paramétrica utilizada para clasificación y regresión. El enfoque de interpretar un árbol de decsión se puede comprender directamente en un conjunto de reglas si-entonces simples. De ese modo, la representación hace que el lector pueda interpretar el resultado y sea fácil de entender. Esta sección presenta los aspectos básicos características del método del árbol de decisión para la clasificación [42], [33], [43].

Representación

En 1986, Quinlan introdujo un algoritmo para inducir árboles de decisión ID3 [44]. El ID3 El algoritmo se mejoró en 1993 y se actualizó a C4.5 [45]. El árbol de decisiones es un codicioso algoritmo. Consiste en divisiones recursivas en sencillos pasos.

Según la descripción de [33], los árboles de decisión consisten en la clasificación de instancias ordenándolas desde la raíz del árbol hasta algún nodo hoja. Normalmente, los datos de entrada se representan como valor-atributo. En cada nodo, se implementa una función de prueba específica para decidir la rama o la hoja para a la que pertenece cada instancia a la clase correspondiente. Este procedimiento se realiza en cada nodo hasta que se encuentre una hoja a la que se le conoce como nodo terminal, entonces la clase predicha del ejemplo dado será su etiqueta [42], [33], [46].

La Figura 2.10, tomada y modificada de [47], ilustra el proceso de predicción para predecir un nuevo solicitante de empleo de acuerdo con su nivel educativo y sus años de experiencia.

Una de las reglas que se pueden extraer del gráfico mostrado en la Figura 2.10 es: SI el solicitante tiene una licenciatura y tiene menos de dos años de experiencia, ENTONCES su desempeño será calificado como deficiente. El árbol de decisión funciona con las clases binarias, pero se puede ampliar para tratar casos de varias clases. Una tarea fundamental de un árbol de decisión consiste en determinar el mejor atributo que divide los datos de manera eficiente en cada etapa. Es considerado un algoritmo codicioso porque busca la propiedad mas importante que sirva como punto de comparación, Grado Académico, en el caso de la Figura 2.10. Esta tarea se lleva a cabo recursivamente desde la raíz hasta el último nodo terminal (Experiencia, para el caso mostrado en la Figura 2.10). En ese sentido, elegir el atributo que sea más útil para clasificar los datos debe tener el grado máximo de discriminación.

La ganancia de información es una técnica estadística que mide qué tan bien el atributo

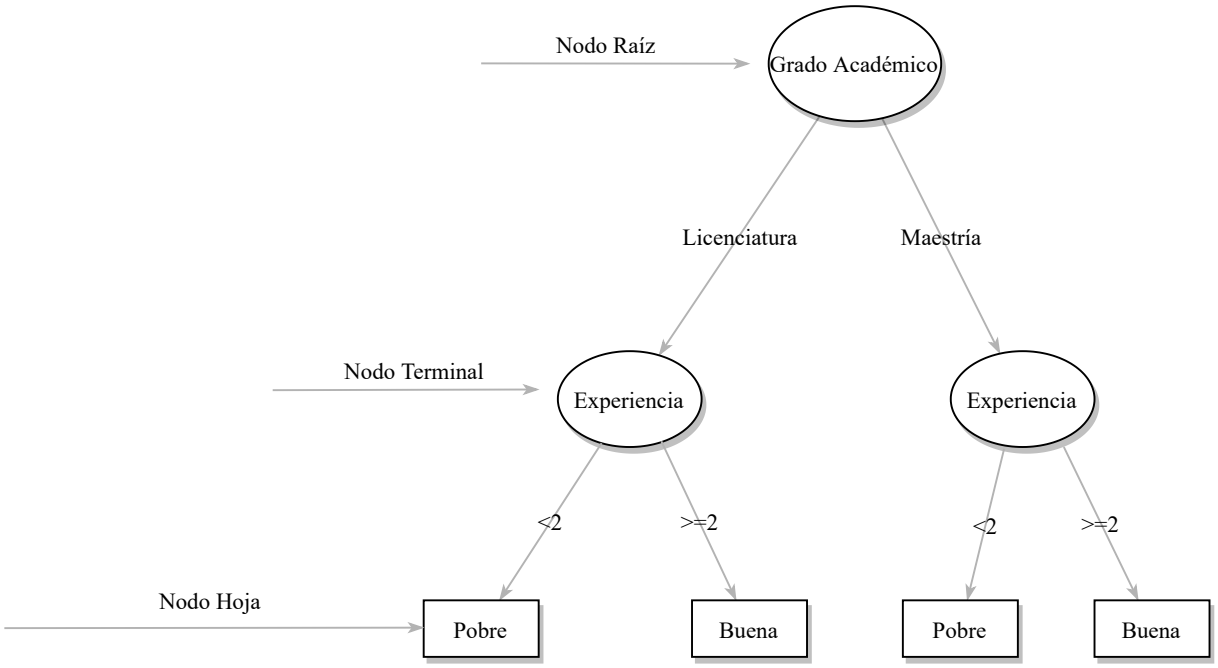


Figura 2.10: Representación gráfica de un árbol de decisión.

divide los datos en un árbol de decisión. Antes de definir este criterio [44], comencemos por introducir la entropía.

La entropía, se define en la teoría de la información, cómo el número mínimo de bits que se necesita para codificar la clase de una instancia. También se denomina medida de impurezas. Asumir que una colección de muestras S y S_m es el número de ejemplos que responde cada nodo m, S_m^j es el número de ejemplos que pertenecen a la clase j en nodo m ilustrado en la ecuación 2.1 con la siguiente representación:

$$\sum_{j=1}^{k} S_m^j = S_m \tag{2.1}$$

La probabilidad de la clase j, dada una instancia x que corresponde a un nodo m se

ilustra en la siguiente ecuación 2.2 como:

$$p(c_j|x,m) = p_m^j = \frac{s_m^j}{s_m} \tag{2.2}$$

El cálculo de la Entropía se define bajo la siguiente ecuación 2.3 como:

$$Entropia(S) = -\sum_{J=1}^{k} p_m^j \log_2 p_m^j \tag{2.3}$$

Si todos los miembros de los ejemplos de entrenamiento pertenecen a la misma clase, por ejemplo, en la Figura 2.10 si todos los solicitantes de empleo tienen maestría, entonces uno de los las probabilidades de clase es igual a 1 y la otra es igual a 0, en este caso la entropía será igual a cero.

Por otro lado, si las instancias se dividen por igual en las dos clases, significa que cualquiera de las probabilidades de clase es igual a 0.5 y la entropía es igual a 1. Cuanto menor es la entropía, más informativo es el atributo [42], [33]. Dada la entropía, la efectividad de un atributo específico puede medirse por criterio de obtención de información de la siguiente manera:

$$Ganancia(S,A) = Entropia(S) - \sum\nolimits_v \in valores(A) \frac{|S_V|}{|S|} Entropia(Sv) \tag{2.4}$$

donde los valores de A indican todos los posibles valores que el atributo A puede tomar, y S_v es el subconjunto de toda la colección complenta de muestras representadas por S para el cual el atributo A toma el valor de v. El primer término de la ecuación es la entropía completa antes de particionar el conjunto de datos y el segundo término de la ecuación es la entropía después de dividir las instancias usando el atributo A Esto significa que la ganancia de información Ganancia(S,A) es la reducción esperada de entropía después de conocer el valor del atributo A [33].

A continuación se mencionan las ventajas y desventajas de los arboles de decisión según [48], [33], se pueden mencionar entre las ventajas:

- Son una herramienta que tiene una representación esquemática simple que incluso puede ser interpretado por no profesionales del area de análisis de datos.
- Un árbol se puede representar por un conjunto de reglas comprensibles para el lector.
- Es una herramienta no paramétrica, por lo que no requiere ninguna especificación de forma funcional.
- El árbol de decisiones puede procesar valores atípicos y faltantes.

  Entre las desventajas:

- El árbol de decisiones puede ser computacionalmente costoso.
- Puede fácilmente sobreajustar los datos, pero en la práctica existen varias herramientas para evitar el sobreajuste, como postpoda y prepoda.
- En la práctica, el árbol de decisiones es menos apropiado para las tareas de estimación en regresión.

Redes Neuronales Artificiales

En el Capítulo 4, se utilizaron redes neuronales artificiales. De ese modo, para dar al lector las nociones básicas del tema, a continuación se presenta la teoría básica. La red neuronal artificial, según[47], es un modelo matemático que intenta simular la funcionalidad de los elementos biológicos tales como el sistema nervioso. Opera con tres reglas básicas: multiplicación, suma y activación y las entradas están ponderadas, lo que significa que cada valor de entrada se multiplica por un valor específico denominado peso. Seguidamente todas las entradas ponderadas se agregarán con un término de sesgo finalmente la suma de todas las entradas ponderadas y el término de sesgo se transformará mediante una función de activación que calcula la salida.

Los pesos que son asociados con cada entrada proporcionan la fuerza de la sinapsis. Cuanto mayor sea el peso asociado con un entrada, más fuerte será la misma. Los pesos

pueden ser positivos o negativos, cuando el primer caso (w_i>0), eso indica una conexión excitadora, mientras que un peso negativo inhibe la actividad neuronal.

El elemento básico de procesamiento se llama perceptrón que tiene entradas x_i provenientes del entorno como una entrada externa o puede ser el resultado de otros perceptrones. Para derivar la salida de estos últimos, se puede lograr con la siguiente ecuación 2.5 mostrada a continuación:

$$y = \sum_{i=1}^{N} w_i x_i + b \tag{2.5}$$

Donde b es el término de sesgo y también se denomina umbral de la neurona. Además, se puede considerar una entrada adicional, b siempre es la unidad y su peso . En este caso, la salida del perceptrón se puede escribir como un producto punto:

$$y = w^T x \tag{2.6}$$

Donde w y x son dos vectores. La función de activación o la función de transferencia define las propiedades del neurona artificial y podría ser cualquier función de activación $\phi(.)$ y la salida será como la siguiente forma:

$$y = \phi(\sum_{i=1}^{n} w_i x_i + b) \tag{2.7}$$

La función de activación se puede determinar en según el problema que necesite solucionar la neurona artificial. Eso actúa como una entidad de transformación de modo que la salida de una neurona toma un valor entre cierto rango por ejemplo:

$$[0, 1]|[-1, 1] \tag{2.8}$$

Las funciones de activación más populares son: umbral, también es llamada escalonada que tiene solo dos resultados posibles (cero o uno). Según [49] y [50], toma la siguiente

forma:

$$y = \left\{ \begin{array}{l} 1, w^T x \geq b \\ 0, w^T x < b \end{array} \right\} \tag{2.9}$$

La función sigmoidea es la más utilizada y es no lineal. La función sigmoidea se define de forma creciente y que exhibe equilibrio entre el caso lineal y no lineal cuya gráfica es en forma de "S". Según [49] y [50], toma la siguiente forma:

$$y = sigmoide(w^T x) = \frac{1}{1 + \exp(-w^T x)} \tag{2.10}$$

La función sigmoidea toma valores entre el rango cero y uno, pero en algunos modelos, es beneficioso usar el intervalo [-1,1]. En el último caso, el umbral se puede definir como:

$$y = \left\{ \begin{array}{l} 1, w^T x \geq b \\ 0, w^T x = b \\ -1, w^T x < b \end{array} \right\} \tag{2.11}$$

Entonces, todo el marco matemático sobre cómo fluye la información desde la entrada para producir la salida puede ser representado como muestra la Figura 2.11 tomada de [47]:

Fortalezas y debilidades

Según [47], las fortalezas y debilidades de la red neuronal artificial según [51] son las siguientes:

Ventajas:

- Se puede utilizar para resolver problemas de programación lineal y no lineal.
- No hay conocimiento previo del proceso relacionado con los datos generados para que la red neuronal artificial pueda ser utilizada.

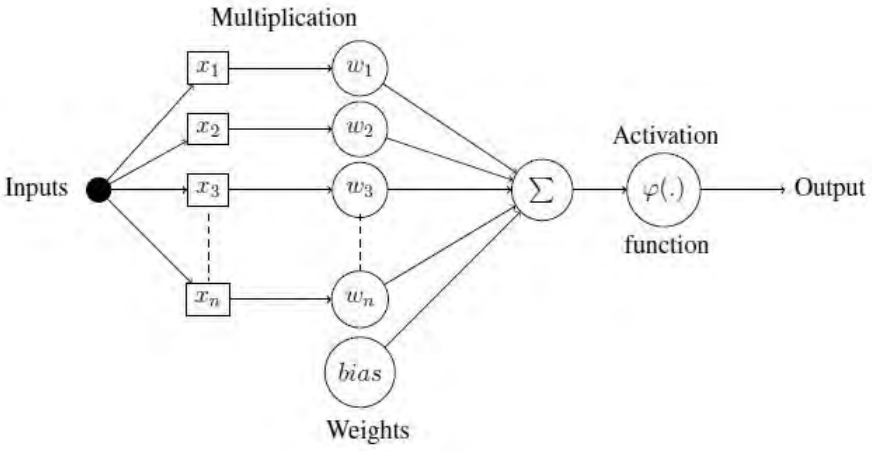


Figura 2.11: Representación gráfica de red neuronal de tres capas.

- La capacidad de aprender de los ejemplos proporcionados hace la red neuronal un algoritmo que aprende y no es necesario reprogramarla.
- Ha tenido éxito para resolver muchos problemas de clasificación, agrupamiento y regresión.

Desventajas:

- Nunca debe verse como una panacea para todos los problemas del mundo real, porque otras técnicas son poderosas en su propio contexto.
- El éxito del modelo depende de la cantidad de datos.
- Falta de pautas claras para las cuales la arquitectura de red artificial es óptima porque el proceso involucra prueba y error.

Las redes neuronales artificiales y los anteriores algoritmos de aprendizaje supervisado requieren participación activa de un experto del dominio de aplicación. Es decir, en un caso de negocio es indispensable un experto que conozca la correspondencia de entradas y salidas de un proceso específico.

Algoritmos de aprendizaje no supervisado

En esta sección se presenta el algoritmo de aprendizaje no supervisado *Kmeans*, el cual se enfoca en la clasificación y segmentación de los datos, los cuales no requieren estar etiquetados por el usuario y también han sido utilizados para detección de anomalías. De ese modo, se menciona que existen una variedad de este tipo de algoritmos si el lector por su cuenta está interesado en profundizar sobre los mismos.

Algoritmo k-means

En el presente trabajo algoritmo de aprendizaje no supervisado de segmentación en k grupos (kmeans, por sus siglas en inglés), fue utilizado para validar un caso de prueba relacionado con compras en línea en el capítulo 3. Por consiguiente, se aborda la teoría básica de dicho tópico para dar una idea general al lector del algoritmo y sus características. Según [31], es un método de cuantización vectorial, originalmente fue diseñado para la aplicación en el procesamiento de señales [52]. Actualmente el enfoque de k-means es definir una cantidad de n grupos o particiones de un conjunto de datos, el objetivo es realizar muestras o agrupaciones con atributos similares para ser analizados de forma independiente. El criterio de agrupación de estos datos es la media más cercana de la distancia entre dos puntos de la muestra de datos. Las principales aplicaciones de este método en la industria de procesos son para dividir los datos del proceso en varios modos de operación, diferentes tipos de fallas o diferentes grados de productos.

Entre las aplicaciones industriales se encuentran:

- [53], Utilizó la agrupación de k-means para derivar la regla de segmentación del subespacio variable para el monitoreo de un proceso por lotes.
- [54]. Implementó k-means para construir el modelo de predicción de pendiente en una aplicación de red de sensores inalámbricos.

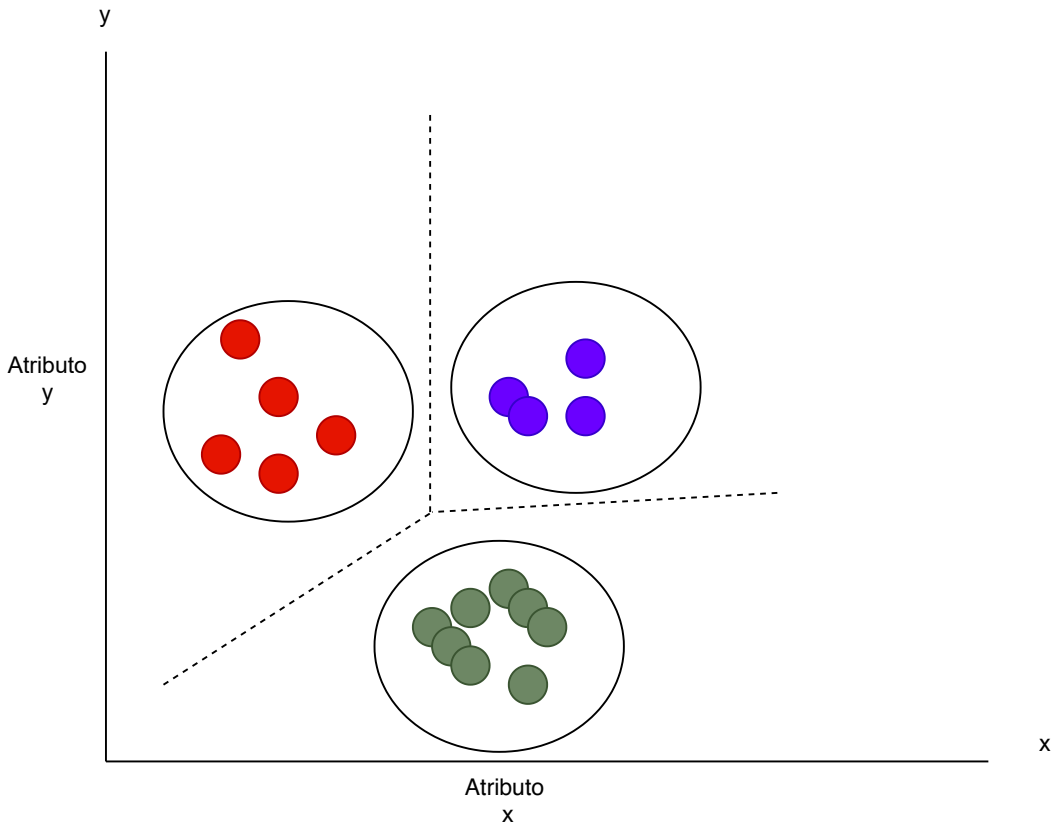


Figura 2.12: Representación gráfica k-means.

En la Figura 2.12, los puntos de colores representan datos agrupados con propiedades o atributos similares que son cuantificados a través de los eje x y y, las lineas punteadas son para dividir los distintos grupos de datos. Para un caso práctico estos datos podrían representar dos variables: el nivel de calidad de un material y el costo del mismo, aplicando este algoritmo se pueden crear asociaciones y agrupar los materiales por su calidad según su nivel de costo.

Analítica de datos y negocio

Según [55], el ciclo de un proyecto de ciencia de datos involucra varias actividades desde el punto de vista del negocio. De ese modo, es importante describir al lector dichas acciones para que tenga en consideración las capacidades que un clúster como el realizado en este trabajo debe desplegar para dar soporte a procesos clave como: adquisición, almacenamiento, procesamiento y analítica de la información en un entorno de producción. En la Figura 2.13 se describen las siguientes actividades [55]:

- Problema de negocio. Un ejemplo pueden ser las ventas de una determinada empresa que tiene potenciales compradores que visitan el sitio web del negocio pero no comprando sus productos. Este planteamiento puede indicar que se necesita un estudio para predecir la tendencia sobre consumo basado en preferencias del cliente y como desea que le entreguen el servicio para que el pueda realizar el consumo en el sitio de ventas del negocio.
- Identificar los datos involucrados: en esta fase se analiza que tipos de datos y sus atributos, es decir, calidad, cantidad y formato. Además, los orígenes de los mismos, tales como: almacenes de datos, archivos de bitácora, documentos de internet, datos generados por sensores y dispositivos de red. De ese modo, esta primera identificación, facilita la selección del tipo de herramientas de software para la adquisición de datos.
- Adquisición de datos. En el punto anterior se reconocen los tipos, fuentes y calidad

de datos disponibles. Por consiguiente, los componentes de software para adquirir datos son utilizados en esta etapa que tiene por objetivo almacenar datos de forma masiva o según se requiera, en lote, por flujo en tiempo real, solo por mencionar algunos casos.

- Hipótesis y Modelado. En esta parte del ciclo se plantean preguntas tales como: ¿Qué está pasando?, ¿Por qué está pasando?, ¿Desde cuándo esta pasando? y ¿Qué pasará?. Es decir, en esta actividad se construyen visualizaciones y modelos predictivos. De ese modo, en esta parte del ciclo, se seleccionan las herramientas que incluyen algoritmos de aprendizaje automático, dicho sea de paso, existen una amplia variedad incluida en distintos *frameworks*.
- Medir la efectividad de la solución. Como ya se mencionó, los modelos y visualizaciones sirven de apoyo para dar respuestas al problema del negocio. Sin embargo, esta etapa tiene por objetivo validar con la realidad los modelos producidos basados en los datos analizados.
- Realizar mejoras. Cada modelo ofrece un porcentaje de confianza que, en algunos casos, no es suficiente para resolver el problema del negocio. Sin embargo, es necesario considerar la posibilidad de construir más de un modelo y seleccionar el mejor.
- Comunicación. En esta etapa se informa de las posibilidades del modelo, así también, las debilidades y beneficios del mismo.

Modelización de los datos

En está subsección se presenta una metodología general de [31] y [38]. Para construir modelos predictivos que son los encargados de encontrar nuevos patrones y significados de la información previamente hay que ejecutar una serie de actividades y son las siguientes:

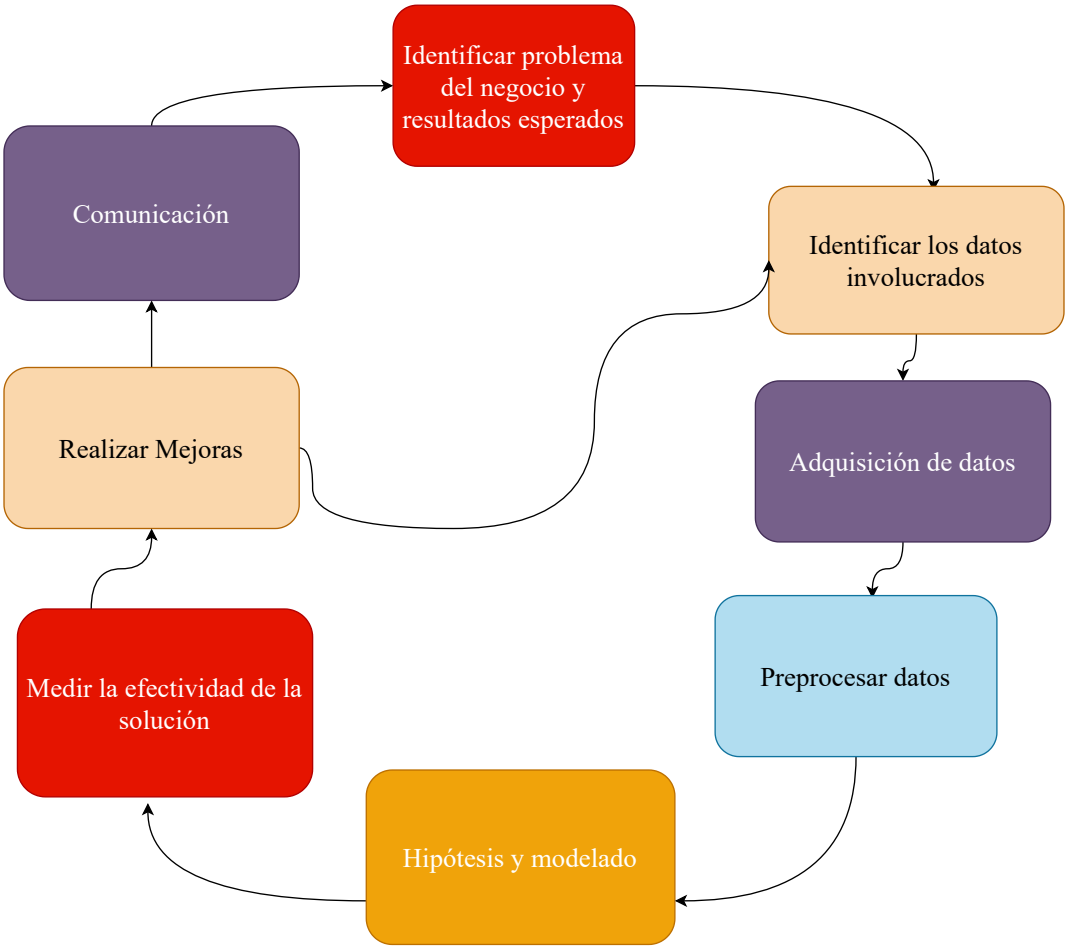


Figura 2.13: Ciclo de un proyecto de ciencia de datos.

Preparación de los datos

Atender el proceso de preparación de datos dentro de una organización es esencial para construir modelos exitosos que suelen generar un beneficio a largo plazo para el negocio. Asimismo, el conocimiento de qué datos existen y cómo se pueden combinar con otras fuentes de datos le proporcionará información que no era posible visualizar anteriormente.

Análisis exploratorio inicial de los datos

Este es el paso en el que comprende sus datos y comienza a adquirir intuición sobre las relaciones entre las variables. Esta exploración se realiza mejor con un experto en el dominio, es decir, un experto de la industria o campo donde se busca implementar la solución derivada del análisis de datos.

Preprocesamiento

Esta tarea se relaciona con mejorar la calidad de los datos. Además, es posible que se necesiten algunas transformaciones y cambios en los mismos que faciliten y mejoren la selección del modelo de una manera más eficiente.

Selección del modelo, entrenamiento y evaluación del desempeño

De manera general, al abordar algún problema práctico, normalmente podemos pensar en varios algoritmos que pueden producir una buena solución, cada uno de los cuales puede tener varios parámetros de entrada. Sin embargo, existen dos preguntas elementales antes de elegir un algoritmo, por ejemplo:

1. ¿Cómo elegir el mejor algoritmo para determinado problema?
2. ¿Cómo configurar los parámetros para mejores resultados?

Esta actividad se conoce como selección de modelo [56]. Para ilustrar la tarea de selección del modelo, considere el problema del ajuste de curva.

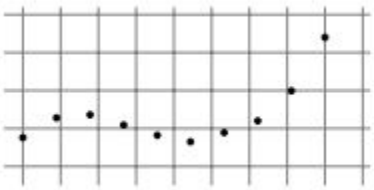


Figura 2.14: Conjunto de datos.

Podemos considerar ajustar un polinomio a un conjunto de datos de interés como los mostrados en la Figura 2.14 tomada y modificada de [56]. Sin embargo, el problema es determinar el grado para obtener mejores resultados para el conjunto de datos: un grado pequeño puede no ajustarse bien a los datos (es decir, tienen un gran error de aproximación), mientras que un alto grado puede conducir a un sobreajuste, es decir, tendrá un gran error de estimación. A continuación, se representa el resultado de ajustar un polinomio de grados 2, 3 y 10. Puede observar que, el riesgo empírico (tasa de fallas en un intervalo) disminuye a medida que aumentamos el grado. Sin embargo, al observar los gráficos de la Figura 2.15 tomada y modificada de [56], se puede intuir que establecer el grado en tres, ver Figura 2.15(2) puede ser mejor que en diez, ver Figura 2.15(3). De ello se deduce que el riesgo empírico por sí solo no es suficiente, en la selección del modelo normalmente la intuición y experiencia puede conllevar al error frecuente si la persona no tiene un historial que lo avale con buenos resultados en la práctica.

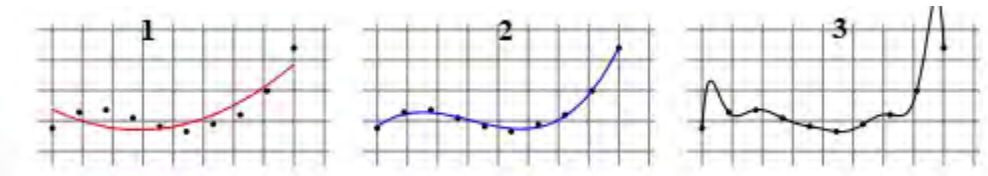


Figura 2.15: Ajuste polinomial de conjunto de datos.

El ejemplo anterior se conoce como paradigma de minimización de riesgos estructurales

(SRM, por sus siglas en inglés). Este es factible cuando un parámetro controla la compensación sesgo-complejidad, tal como el grado del polinomio ajustado en el ejemplo anterior. El segundo enfoque se basa en el concepto de validación. La idea básica es dividir el conjunto de entrenamiento en dos. Uno se utiliza para entrenar cada uno de los modelos candidatos y el segundo se utiliza para decidir cuál de ellos produce los mejores resultados.

En resumen, *Big Data* en la detección de anomalías abarca varios sectores de la economía, tales como: finanzas, seguridad, transporte, manufactura, educación y ámbitos militares, por mencionar algunos. Cabe destacar, su crecimiento uso y difusión es justificado por el costo accesible de las capacidades computacionales actuales.

Principalmente por el cómputo en nube y el crecimiento exponencial de los datos. Sin embargo, hay que recordar, la extracción de valor en la información se debe al uso de métodos actuales de *Machine Learning*, por lo que los avances en esta rama de la inteligencia artificial, son cruciales en la construcción de algoritmos predictivos derivados del análisis masivo de datos. La siguiente sección, se enfoca en las herramientas y componentes de software utilizados en este trabajo para implementar una plataforma como servicio para el análisis *Big Data*.

2.2. Marco Tecnológico

Esta sección tiene por objeto introducir al lector con los conceptos y teorías, enfoques y aproximaciones relacionados con el conjunto de componentes y tecnologías de software necesarios en la integración de una plataforma de análisis masivo de datos para detección de anomalías.

Se presentan los siguientes temas: plataformas *Big Data* y ecosistemas, Mesos como administrador de clúster, sistema de coordinación distribuida ZooKeeper, Hadoop para almace-

namiento masivo de datos, cómputo paralelo e iterativo con Spark, componentes de lectura, adquisición y escritura de datos en lote y por flujo de datos. Además, se incluyen otras herramientas necesarias para análisis y visualización de de datos.

2.2.1. Plataformas y Ecosistemas Big Data

Según [12],[57],[58], el uso de plataformas como servicio para el análisis masivo de datos, actualmente está vinculado al paradigma de cómputo en la nube, debido a que las industrias pagan por lo que consumen y ahorran en aspectos relacionados con la infraestructura computacional, tales como: mantenimiento, consumo energético, impuestos y espacio físico. En resumen, los criterios a tener en cuenta por parte de una organización para el uso y selección de un PaaS con fines de análisis masivo de datos, se centra en tres aspectos: compatibilidad con los componentes de software más utilizados en el mercado, disponibilidad financiera para licenciamiento y escalabilidad.

BigML

Es una plataforma pionera en el servicio de aprendizaje automático como servicio (MLAAS, por sus siglas en inglés). Con una interfaz web intuitiva BigML pone a disposición de sus usuarios los recursos de su plataforma por medio de cualquier navegador moderno de internet.

Sin embargo, un componente adicional es su interfaz de programación de aplicaciones API (por sus siglas en inglés) y la arquitectura de transferencia de estado representacional (REST, por sus siglas en inglés). Además, la plataforma cuenta con una multitud de bibliotecas y herramientas que facilitan las actividades relacionadas con el análisis de datos [59]. En la Figura 2.16, se ilustra el logo de la plataforma BigML tomado de [59], utilizado como símbolo comercial y representante de su producto.

OpenShift

Figura 2.16: Logo de BigML.

Es una plataforma de gestión de contenedores optimizada para aplicaciones web. Desarrollada por la firma Red Hat , permite la construcción, pruebas, y despliegue de aplicaciones web sin depender de servidores dedicados para cada aplicación. Es decir, OpenShift ejecuta aplicaciones en contenedores Docker, citando a [21]:

> Los contenedores Docker son una abstracción de una pieza de software empaquetada en un completo sistema de archivos que contienen todo lo necesario como: código, tiempo de ejecución, herramientas y bibliotecas del sistema operativo. En resumen todo lo que se puede instalar en un servidor. Esto garantiza que siempre funcionará adecuadamente, sin importar el entorno en el cual está ejecutandose

Parte de la arquitectura de esta plataforma se diseñó para integrar la compatibilidad con el administrador kubernetes [2] . Este último provee el núcleo funcional de orquestación de contenedores, incluyendo características como tolerancia a fallas, monitoreo y reprogramación automática de contenedores y escalamiento horizontal. Además, tiene soporte *Multitenant*.

En la Figura 2.17, tomada y modificada de [60], se puede apreciar la consola de administración de la plataforma, en su versión 4, en la cual se pueden administrar: proyectos, usuarios, eventos, cargas de trabajo, redes, monitoreo y cómputo, entre otros.

DCOS

Distributed Cloud Operating System (por sus siglas en inglés), está basado en un núcleo de sistemas distribuidos de nombre Mesos, el cual se describe en la siguiente sección

[2] A diferencia de Docker, kubernetes esta diseñado para ejecución en clúster y no en un solo nodo como podria ser el caso de Docker que se ejecuta en ambos modos

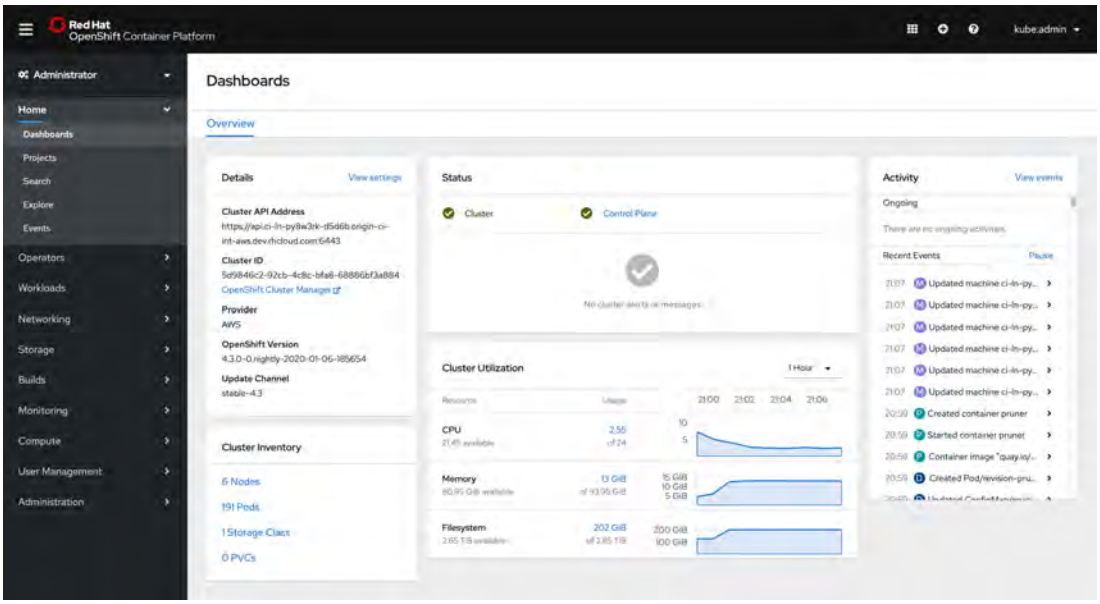


Figura 2.17: Vista de la consola de recursos OpenShift.

detalladamente. DCOS permite administrar múltiples unidades de cómputo que de manera conjunta conforman una supercomputadora, esto se logra al sumar los recursos de hardware de los mencionados.

Además, gestiona la asignación de procesos de forma automatizada, facilitando la comunicación entre ellos. de ese modo, simplifica la instalación y gestión de servicios distribuidos. Su interfaz web incluida y la interfaz de línea de comandos (CLI), facilitan la administración remota y el monitoreo de sus servicios. Los requisitos para la instalación mínima de un clúster DCOS, se compone de cinco nodos, ver Figura 2.18; clasificados en maestros y agentes. Estos últimos se dividen en privados y públicos. Además, un nodo de arranque (Bootstrap, por su nomenclatura en inglés), estos se describen con respecto a su funcionalidad como:

Nodo Maestro

Son los coordinadores y encargados de gestionar las tareas asignadas a los nodos agentes.

Agentes privados

Ejecutan tareas asignadas por nodo maestro. Además, alojan los contenedores Docker y servicios que se implementan en el clúster.

Agentes públicos

Proporcionan acceso a través de Internet a una red interna a los servicios en el clúster a través de balanceadores de carga de trabajo. Sin embargo, tambien pueden ejecutar tareas asignadas por los nodos maestros.

Nodo Bootstrap

Arranque separado para la instalación de DCOS y los archivos de actualización.

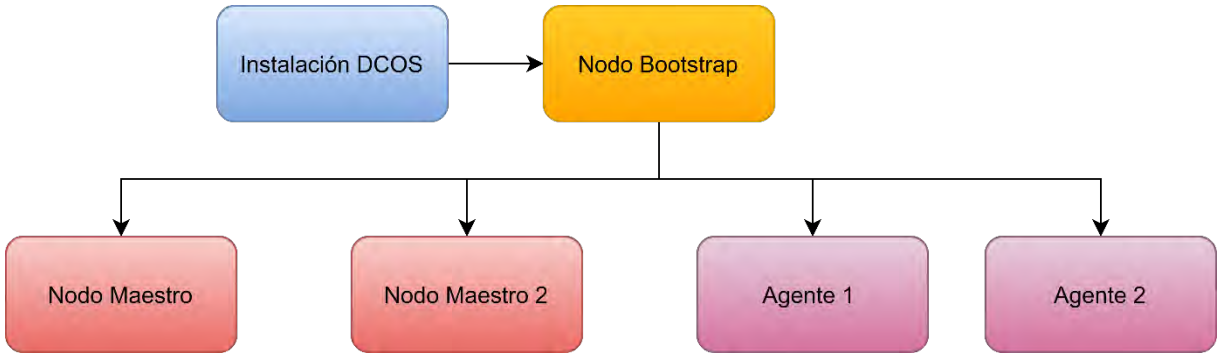


Figura 2.18: Nodos mínimos que integran un clúster DCOS.

Descripción de nodo Bootstrap

En este nodo es importante mencionar los dos siguientes detalles:

- El nodo Bootstrap solo se usa durante el proceso de instalación y actualización. De ese modo, no hay recomendaciones específicas para el almacenamiento de alto rendimiento o los puntos de montaje separados.
- La instalación de DCOS se ejecuta en un solo nodo Bootstrap con dos núcleos de procesador, 16 GB de RAM y 60 GB de disco duro.

Recomendaciones para uso en producción

Para implementaciones críticas de negocios, [61] recomienda utilizar tres nodos maestros en lugar de uno, estos últimos por lo general gestionan múltiples cargas de trabajo mixtas. Se espera que estas últimas estén continuamente disponibles ya que son consideradas críticas para el negocio.

En ese sentido, es importante mantener la información de los registros de los trabajos en curso usando los registros replicados de Mesos y ZooKeeper, dos tecnologías de clúster encargadas de gestionar y coordinar tareas. La sincronización periódica de los registros se puede realizar con la utileria *fsync*, encargada de escribir los datos temporales del sistema operativo en un formato de disco. Sin embargo, esta actividad puede generar entradas y salidas aleatorias que son costosas en tiempo de acceso. Por lo que [61] recomienda lo siguiente:

Conectar localmente al hardware de cómputo: unidades de almacenamiento de estado sólido SSD (por sus siglas en inglés) o dispositivos de memoria no volátil express NVME (por sus siglas en inglés). De tal manera, esto permite reducir la probabilidad de problemas relacionados a latencia, para este caso, se refiere en al tiempo consumido para la lectura de un archivo de E/S.

Generalmente, se recomienda utilizar los NVME para sistemas de archivos que alojan registros replicados de nodo maestro [61]. Al planificar los requisitos de almacenamiento, se debe evitar usar un solo dispositivo en red para ello. Para conectarse a los nodos del clúster, evitar dispositivos como: Redes de Área de Almacenamiento SAN (por sus siglas en inglés) y Sistemas de Archivos en Red NFS (por sus siglas en inglés). Este tipo de arquitectura presenta una mayor posibilidad de latencia comparado con el uso de almacenamiento local.
Un punto importante a mencionar es que al utilizar un clúster DCOS se debe tener en

Requisitos	Mínimos	Recomendados
Nodos	1	3 a 5
CPU	4 núcleos	4 núcleos
Memoria	32 GB RAM	32 GB RAM
Memoria en disco	120 GB	120 GB

Tabla 2.1: Requisitos de un nodo maestro.

Requisitos	Mínimos	Recomendados
Nodos	1	6 o más
CPU	2 núcleos	2 núcleos
Memoria	16 GB RAM	16 GB RAM
Memoria en disco	60 GB	60 GB

Tabla 2.2: Requisitos de un nodo agente.

consideración la latencia de la red y los problemas de ancho de banda. Estos pueden ocasionar que las sesiones del cliente excedan el tiempo de espera y afectar negativamente el rendimiento y la confiabilidad del clúster DCOS.

Puntos de montaje

Según [61], recomienda establecer ciertos puntos de montaje por separado si el clúster tendrá constantes cargas de trabajo. En la Tabla 2.3, se describen las rutas de los directorios sugeridas para lo anteriormente mencionado.

En forma resumida, en esta sección abordamos dos escenarios interesantes recomendados por [61], previos a cualquier instalación de un clúster de producción DCOS. El primero es implementar puntos de montajes separados y memoria en disco de alta velocidad, esto si tenemos un negocio con múltiples y constantes cargas de trabajo, así como una frecuente demanda de datos.

El segundo escenario es hacer uso de un hardware convencional y configuración por defecto del clúster DCOS, cuando no requerimos un modelo de negocio con las exigencias

Punto de montaje	Descripción
/var/lib/dcos	Si la implementación de clúster será integrado por cientos de nodos y con una alta tasa de cargas de trabajo, se recomienda aislar un directorio al almacenamiento SSD dedicado en un dispositivo separado.
/var/lib/dcos/mesos/master	Directorios de sesión.
/var/lib/dcos/exec	Archivos temporales de servicios DCOS.
/var/lib/dcos/exhibitor	Base de datos Zookeeper
/var/lib/dcos/exhibitor/zookeeper/transactions	Los registros de transacciones de ZooKeeper son muy sensibles a los retrasos en las escrituras en disco. Debe haber un mínimo de 2 GB disponibles para estos registros.

Tabla 2.3: Puntos de montaje DCOS.

mencionadas en el primer caso, por ejemplo, casos de experimentación y con motivos de académicos. La siguiente subsección describe el kernel distribuido Apache Mesos, que es el núcleo base del sistema DCOS.

En la Figura 2.19, se ilustra la consola de administración de la plataforma DCOS. En ella se pueden administrar: paquetes, eventos, cargas de trabajo, redes, monitoreo, cómputo, entre otros. Es muy similar a la consola de OpenShift. La siguiente subsección aborda el núcleo que permite ejecutar cómputo distribuido en DCOS.

Definición de ecosistema Big Data

Es un término que se refiere a la diversidad de aplicaciones y componentes tecnológicos de software que han sido agrupados por el tipo industria en donde pueden ser utilizados y por el tipo de solución que brinda. Algunos son propietarios y su uso se obtiene bajo un licenciamiento autorizado de la empresa desarrolladora. Los componentes de código abierto manejan normalmente otro tipo de licenciamientos, más flexibles para su utilización y distribución.

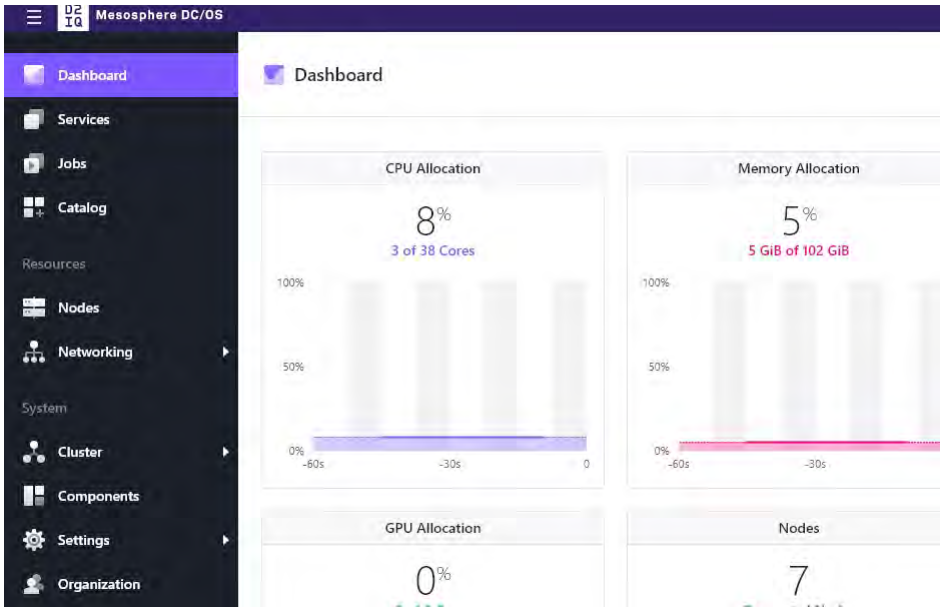


Figura 2.19: Vista de la consola de recursos DCOS.

De tal manera que, entre las implementaciones más importantes para *Big Data* podemos mencionar la de la fundación Apache, sin costo de licenciamiento y algunas propietarias, como la de la empresa Oracle . Sin embargo, el producto final de esos ecosistemas es una combinación de tecnologías, las cuales funcionan como un solo sistema que representa un solución integral para el procesamiento de grandes volúmenes de datos [62]. En la Figura 2.20, tomada y modificada de [63], se muestran algunos de los componentes hasta la fecha del 2019. Actualmente se cuenta con un aproximado de mas de ochocientos componentes de software relacionados con *Big Data*.

Un ecosistema se compone de tres procesos fundamentales [64], tales como:

1. **Ingestión de datos**. Servicios en línea y dispositivos conectados a diversas fuentes proporcionan la fuente de datos.
2. **Analítica de datos**. Este componente se refiere al procesamiento y análisis de datos a gran escala.

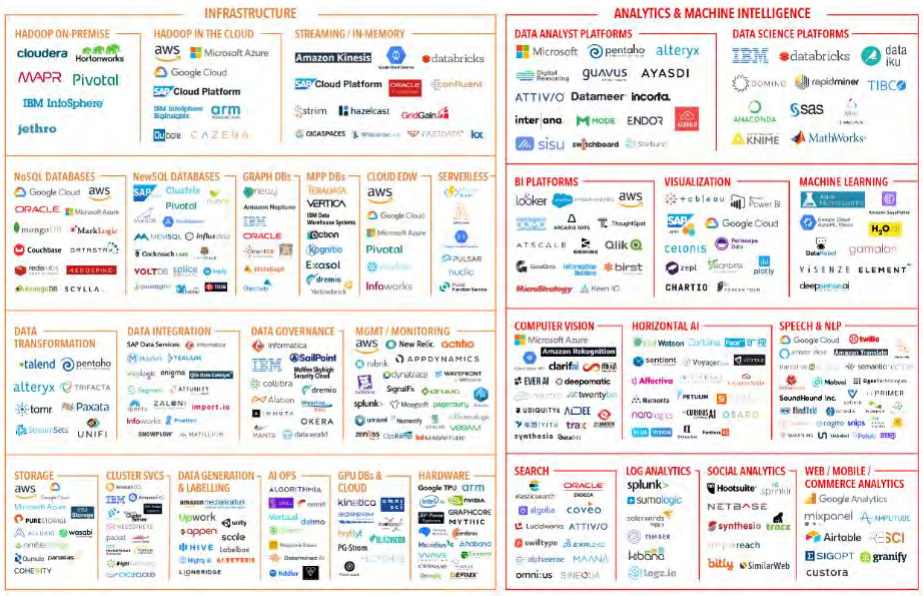


Figura 2.20: Paisaje de tecnologías *Big Data*.

3. **Almacenamiento**. Las bases de datos NoSQL cumplen una función primordial que es almacenar la información procesada.

En la Figura 2.21, tomada de [64] se ilustran las capas que componen un ecosistema *Big Data* que ya se mencionó. Además, otras dos capas adicionales se pueden ilustrar en la misma Figura. La capa de recursos, se refiere a las tecnologías que soportan el ecosistema *Big Data* y la capa de aplicaciones que se orienta al tipo de industria donde se puede aplicar el análisis de datos para obtener un beneficio directo para un impacto positivo en la organización.

Ecosistema Hadoop

Diseñado por la fundación Apache, contiene un conjunto de componentes para el proceso de minería de datos en cantidades masivas, tales como: el administrador de clusters Yarn. Este permite ejecutar y coordinar tareas en un entorno distribuido, además, se encarga de la administración de recursos computacionales que integran el clúster. En la Figura 2.22, se ilustra una arquitectura Big Data. Los dos primeros niveles (HDFS

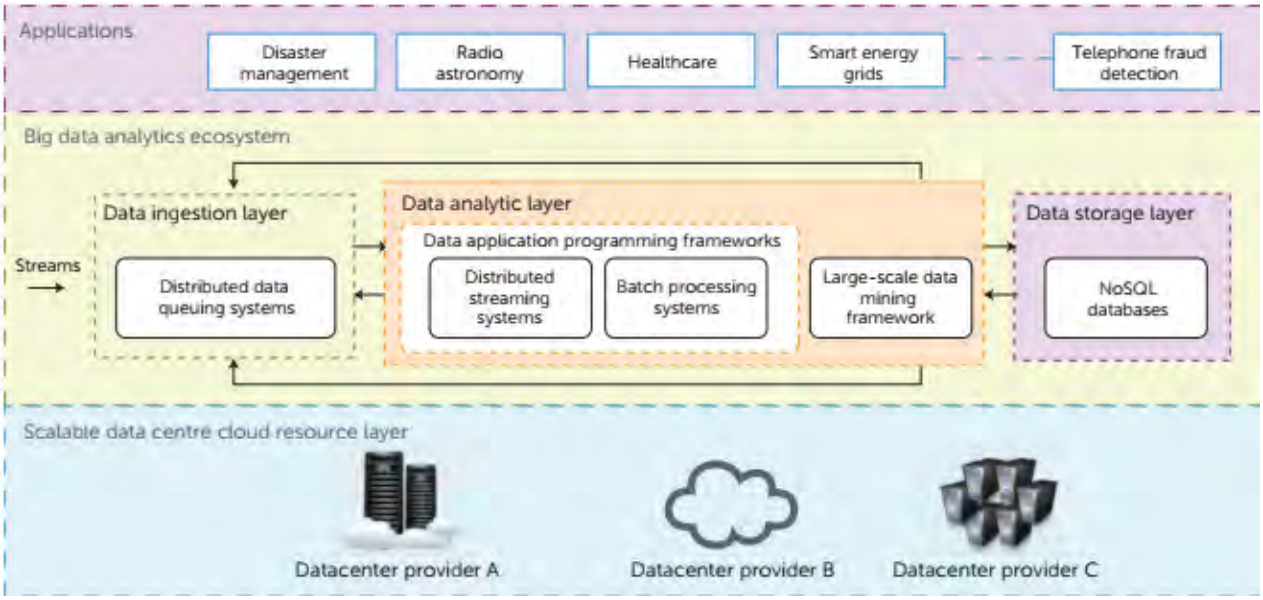


Figura 2.21: Arquitectura general de un Ecosistema *Big Data*.

y Administrador de Recursos Mesos/Yarn, ver Figura 2.22), se refieren al sistema de archivos distribuido de Hadoop y los segundos a los administradores de clúster.

En el tercer nivel, se ilustran componentes de bases de datos NoSQL y el cuarto se compone de las herramientas para procesamiento de datos. Los últimos dos niveles al conjunto de librerías para análisis de datos con *Machine Learning*. La parte final del ecosistema, se conforma por las librerías y entorno que comunica los componentes en el ecosistema para desarrollo de aplicaciones (ver Figura 2.22 los rectángulos en posición vertical más a la izquierda).

2.2.2. Componentes de gestión y administración de clúster

Los entornos actuales de cómputo distribuido utilizan de base hardware económico interconectado por redes informáticas y requieren componentes de gestión de recursos físicos. Por consiguiente, dichos gestores de recursos son el equivalente a un sistema operativo pero

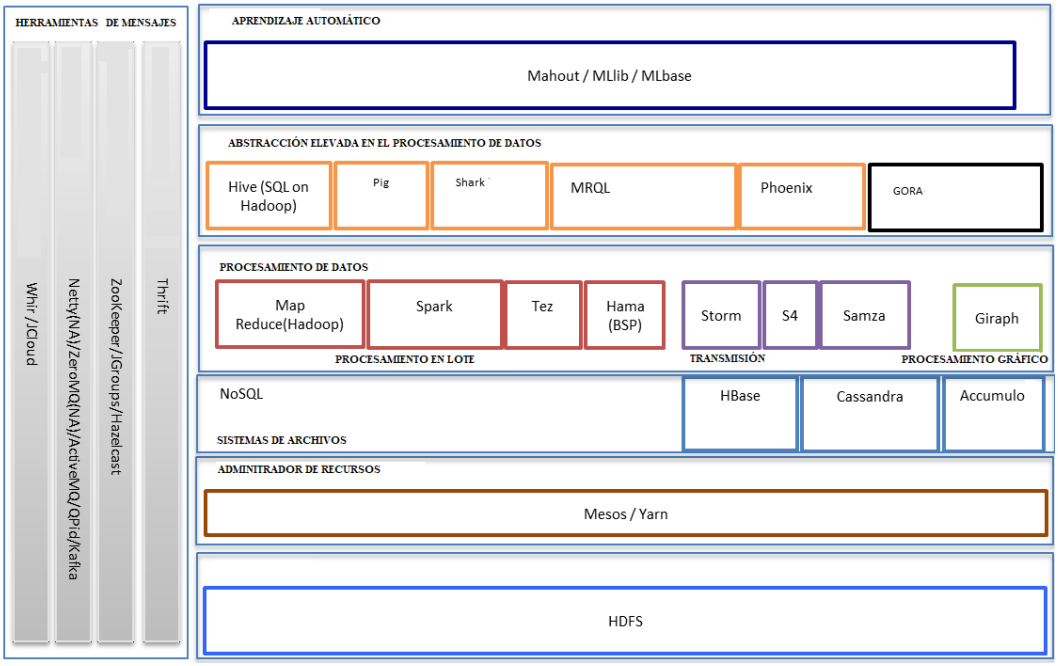


Figura 2.22: Arquitectura Ecosistema Big Data de Apache.

desplegado para administrar y comunicar múltiples equipos de cómputo.

Mesos

Según [58], es un administrador de clúster que está enfocado en la utilización optimizada de recursos compartidos a través de un método dinámico en múltiples *Frameworks*. Mesos surgió en la universidad de California, Berkely en el 2009, y actualmente se utiliza por compañías como Twitter y Airbnb.

Además, la capacidad de compartir recursos disponibles entre nodos (equipos de cómputo, servidores) a través distintas clases de tareas, logra una capacidad de balance y distribución más eficiente de la infraestructura.

Como se muestra en la Figura 2.23, Mesos (1) y (2) se ubica sobre la capa del hardware representada por el centro de datos (0). Por lo que en la literatura le han acuñado el término de *kernel distribuido*. Además, proporciona una vista unificada de los recursos

de todos los nodos, y acceso abierto hacia ellos en una manera similar que lo hace el núcleo del sistema operativo de una sola máquina.

Otra característica es la escalabilidad, esta permite agregar nuevos recursos de hardware, añadiendo mayor capacidad computacional. Además, cuenta con un despachador de doble fase para gestionar la ejecución de tareas de las aplicaciones de software y un núcleo para manejo de aplicaciones que requieren un centro de datos. La herramienta Mesos API, permite desplegar un amplio rango de aplicaciones sin traer la información específica de dominio hacia el núcleo de Mesos. En ese sentido, se anulan de los problemas de los despachadores monolíticos tradicionales.

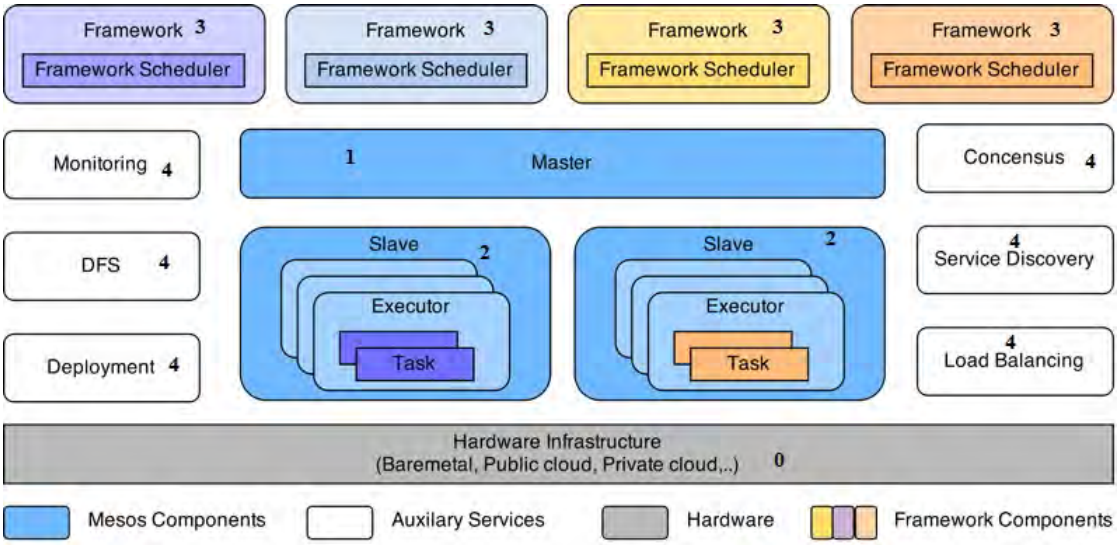


Figura 2.23: Arquitectura Interna Mesos

Nodo maestro

En la Figura 2.23, el Maestro (1), es un nodo responsable de gestionar a otros nodos esclavos (2) y *Frameworks* (3) de herramientas de software. En cualquier momento, el *kernel distribuido* Mesos se encarga de tener un maestro activo seleccionado con la herramienta ZooKeeper, mediante un algoritmo de consenso distribuido.

Si Mesos está configurado para ejecutarse en un modo tolerante a fallas, un maestro se designa a través del protocolo de elección de líder distribuido, y el resto de ellos permanece en modo de espera. De ese modo, es como tener una organización autogestionada de empleados. Por diseño simplificado, el maestro no está destinado a realizar tareas complejas. Este último, ofrece recursos de nodos esclavos a *Frameworks* y de ser aceptados los primeros, reciben instrucciones de tareas que deben ser ejecutadas [58].

Esclavos

Ver Figura 2.23 (0), administran recursos en nodos individuales y están configurados con una política de recursos para reflejar las prioridades del dominio de aplicación (área de negocio importante para el cliente: finanzas, medicina, agricultura, manufactura, aviación, construcción, por mencionar algunos). Los esclavos administran varios recursos, como CPU, memoria, puertos, entre otros, y ejecutan tareas enviadas por *frameworks* [58].

Frameworks

Ver Figura 2.23 (3), son aplicaciones que se ejecutan en Mesos y resuelven un caso de uso específico (almacenamiento de datos y machine learning, por mencionar algunos). Cada marco consta de un planificador y un ejecutor. El primero es responsable de decidir si acepta o rechaza las ofertas de recursos. Los segundos son consumidores de éstos y se despliegan en nodos esclavos y son responsables de ejecutar tareas asignadas [58]. Mesos, es un componente incorporado en DCOS y es considerado cómo uno de los mas eficientes [12].

Sistema de coordinación distribuida ZooKeeper

Es un sistema de coordinación distribuida, desarrollado en Yahoo! y donado a Apache Software Foundation [65]. Por consiguiente, esta clase de componentes son necesarios cuando se requiere la construcción de sistemas distribuidos [66]. Además, incluye un algoritmo de consenso llamado ZooKeeper Atomic Broadcast ZAB (por sus siglas en inglés). De ese modo, se obtiene un servicio centralizado para coordinar aplicaciones

distribuidas. En la Figura 2.24, los esclavos (2), a través de ZooKeeper son informados quien es el Maestro (3). Si el Maestro falla, se asigna un nuevo Maestro de los candidatos existentes (4) y (5).

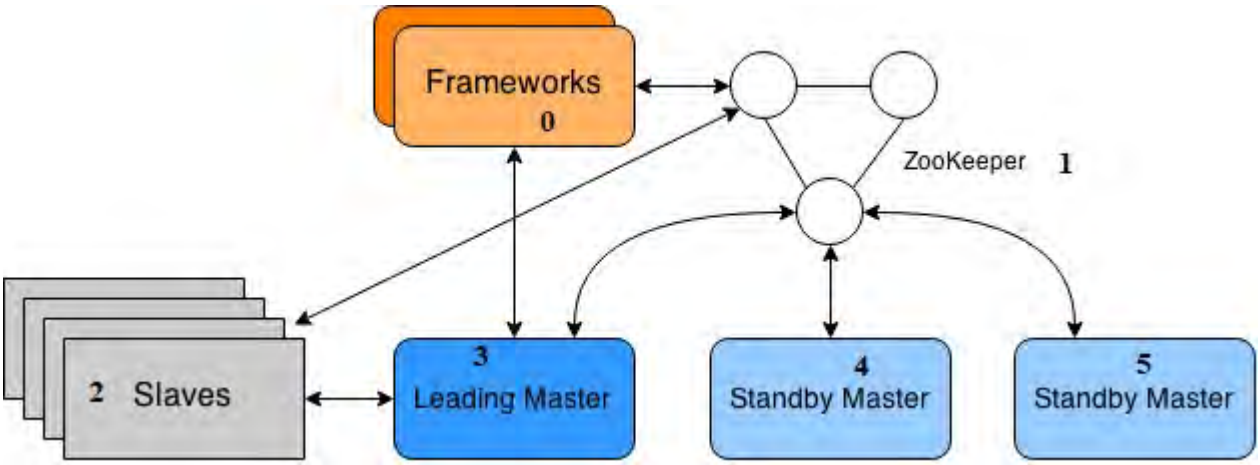


Figura 2.24: Esquema de funcionamiento de Zookeeper.

2.2.3. Componentes de cómputo, almacenamiento y procesamiento de datos

En esta subsección se describen los componentes utilizados para tareas de cómputo, almacenamiento y procesamiento de datos.

Hadoop para almacenamiento masivo de datos

El sistema de archivos distribuidos de Hadoop HDFS (por sus siglas en inglés), proporciona la capacidad de almacenar un gran volumen de datos en hardware básico (haciendo referencia a equipos de cómputo económicos), agrupados como una gran colección de nodos. Además, incorpora las API necesarias para leer estos datos de manera eficiente. De tal manera, HDFS, puede verse como una versión recortada de un sistema de archivos distribuido con énfasis en almacenar grandes datos y leerlos con frecuencia [62],[67],[68].

Además, un componente que viene integrado con Hadoop para realizar cálculos con los datos se le conoce como MapReduce [69] y lo que complementa esta tecnología impulsada por Google [68].

Sistema de Archivos Distribuidos Hadoop

El sistema de archivos distribuido de Hadoop, es utilizado cuando un conjunto de datos supera la capacidad de almacenamiento disponible en una sola máquina física. Por lo tanto, es necesario extenderlo, en varias de estas últimas [67]. Los sistemas de archivos para almacenamiento a través de una red distribuida de máquinas, se denominan sistemas de archivos distribuidos.

Basados en parte, por el uso de las redes de telecomunicaciones, todas las problemáticas de estas se hacen presentes; lo que los hace más complejos que los sistemas de archivos de disco normales. Uno de los mayores desafíos de esta tecnología es hacer que el sistema de archivos tolere fallas de una o varias máquinas, sin sufrir pérdida de datos. HDFS la tecnologia insignia de Hadoop, pero en realidad tiene una abstracción de sistema de archivos de propósito general, por lo que Hadoop se integra con otros mecanismos de almacenamiento, como el de tipo local y Amazon S3 [67].

Arquitectura HDFS

Está diseñado para almacenar archivos de gran volumen, con transmisión de patrones de acceso a los datos, que se ejecutan en grupos de hardware básico [68]. Archivos de cientos de megabytes, gigabytes o terabytes se pueden procesar con Hadoop, sin complicaciones; incluso almacenamiento de petabytes de datos y acceso en tiempo real. La transmisión de acceso a los datos del sistema HDFS se basa en la idea siguiente: el patrón más eficiente de procesamiento es ejecutar la escritura en disco un sola vez, por cada archivo guardado. Sin embargo, para acceder a estos últimos, se permiten múltiples lecturas a las ubicaciones de almacenamiento [67].

Un conjunto de datos que será analizado en el tiempo implicará un largo volumen de

ellos. De ese modo, es más importante acceder al conjunto de datos entero, que propiciar latencia, ocasionada al leer el primer registro y sucesivos de manera secuencial. Los productos básicos Hadoop no requieren hardware costoso y sofisticado. Está diseñado para ejecutarse en clústeres de hardware básico [67], para los cuales la probabilidad de falla en algún nodo es alta, al menos para los clústeres grandes. HDFS está diseñado para continuar trabajando sin una interrupción notable durante fallas en alguno de los nodos. Aunque esto puede cambiar en el futuro, estas son áreas en las que no encaja bien hoy: gestión de datos con baja latencia, dicho de otro modo no puede manejar datos a una alta velocidad, en el rango decenas de milisegundos, no funcionarán bien con HDFS.

Bloques HDFS

Se definen como unidades de almacenamiento de 128 MB. Los archivos se pueden dividir en bloques de tamaño fijo que se almacenan como unidades independientes. A diferencia de un sistema tradicional de archivos de disco, un archivo en HDFS puede ser más pequeño que un solo bloque, y no ocupa el almacenamiento subyacente de este por completo. Por ejemplo, un documento de 1 MB almacenado con un tamaño de bloque de 128 MB usa 1 MB del espacio total disponible [67].

Los bloques encajan bien con la replicación para proporcionar tolerancia y disponibilidad de fallas. Para evitar bloques corruptos, fallas del disco y la máquina, cada uno de ellos se replica en un pequeño número de máquinas físicamente separadas, generalmente tres [67].

Si un bloque deja de estar accesible, se puede leer una copia desde otra ubicación de forma transparente para el cliente. La no disponibilidad de este último, es causada por corrupción o falla de un equipo de cómputo, la solución es reproducir el bloque desde ubicaciones alternativas a otras máquinas, regresando el factor de replicación al nivel normal; dicho de otra manera, volver a tener el mismo número de copias del mencionado. Del mismo modo, algunas aplicaciones pueden optar por establecer un

factor de replicación alto, para los bloques con un archivo frecuentemente usado [67].

Tipos de nodos

Un clúster HDFS tiene dos tipos de nodos que operan en un patrón maestro y esclavo; los primeros denominados *namenode*, haciendo referencia al maestro; los segundos *datanodes* de datos y referentes a los esclavos [67].

Namenodes

Gestionan el espacio de nombres, mantiene el árbol y directorios del sistema de archivos y los metadatos de todos ellos. Esta información se almacena de forma persistente en el disco local en la imagen del espacio de nombres y el registro de edición. El *namenode*, también conoce los nodos de datos en los que se encuentran todos los bloques para un archivo dado. Sin embargo, no almacena ubicaciones de bloque de forma persistente, porque esta información se reconstruye a partir de los *datanodes* cuando se inicia el sistema [67].

DataNodes

Almacenan y recuperan bloques cuando los clientes *namenodes* les asignan dicha tarea, también informan periódicamente a estos últimos las listas de bloques que están guardando. Sin los *namenodes*, el sistema de archivos es inservible. Dicho de otro modo, si la máquina que ejecuta los mencionados falla, todos los archivos se perderían, ya que no habría forma de saber cómo reconstruirlos a partir de los bloques, en los *datanodes*.

Por esta razón, es importante hacer que el *namenode* sea tolerante a fallas, Hadoop proporciona dos mecanismos para esto. El primero es hacer una copia de seguridad de los archivos que conforman el estado persistente de los metadatos, del sistema de archivos.

Hadoop se puede configurar para que el namenode escriba su estado persistente en múltiples sistemas de archivos. Estas escrituras son sincrónicas y atómicas. La opción de configuración frecuente es escribir en el disco local, así como en un montaje NFS

(sistema de archivos de red, por su significado en inglés) remoto.

También es posible ejecutar un *namenode* secundario, su función principal es fusionar periódicamente la imagen del espacio de nombres con el registro de edición para evitar que este se vuelva demasiado grande.

Un *namenode* secundario, generalmente se ejecuta en una máquina física separada, porque requiere considerable CPU (unidad central de procesamiento, por sus siglas en inglés); y tanta memoria, como el nodo de nombre para realizar la fusión. Este ultimo mantiene una copia de la imagen del espacio de nombres combinada, que se puede usar en caso de que falle el nodo de nombre [67].

Mecanismo de funcionamiento Hadoop

El proceso de Hadoop y sus sistema de archivos se ilustra en la Figura 2.25, en la cual se describe el siguiente proceso:

1. El cliente (1A), realiza una petición al Maestro (1B) para escribir un nuevo archivo.
2. Si existe, se envía un mensaje apropiado al cliente. Si no existe, (1B) crea una entrada de metadatos para el nuevo archivo.
3. El archivo que se va a escribir se divide en paquetes de datos (3C) en el extremo del cliente y se crea una cola de datos (4D). Los paquetes de la cola se transmiten luego a los nodos de datos (5E, 6F, 7G) del clúster.
4. La lista de nodos de datos la proporciona el demonio de nodo de nombre (1B), que se prepara en función del factor de replicación de datos configurado. Se crea una canalización para realizar las escrituras en todos los nodos de datos (5E, 6F, 7G) proporcionados por (1B).
5. El primer paquete de la cola de datos se transfiere luego a (5E), el bloque se almacena y luego se copia en (6F) y (7G). La secuencia se repite para todos los paquetes en la cola de datos.
6. Una vez escritos todos los paquetes, el cliente realiza una acción de cierre, lo que

indica que los paquetes en la cola de datos se han transferido por completo.

7. 1A informa 2B que la operación de escritura se ha completado.

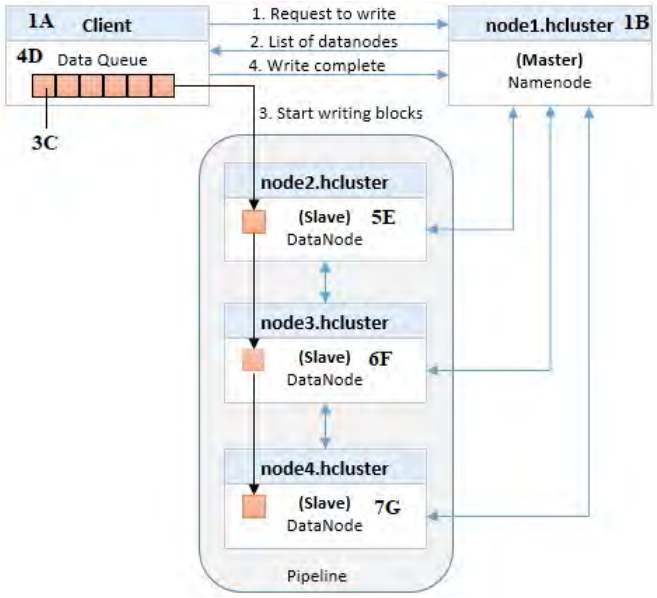


Figura 2.25: Esquema de funcionamiento Hadoop.

Cómputo paralelo e iterativo con Spark

En [70] aparece como: un proyecto de código abierto implementado y mantenido por una comunidad diversa de desarrolladores. Comenzó en 2009 como un proyecto de investigación en el laboratorio de UC Berkeley RAD, para luego convertirse en AMPLab. Previamente se trabajó en Hadoop Map-Reduce y observaron que era ineficiente para trabajos de computación iterativos e interactivos. Por lo tanto, desde el principio, Spark fue diseñado para ser rápido en consultas interactivas y algoritmos iterativos, aportando ideas como soporte para almacenamiento en memoria y recuperación eficiente de fallas. Se publicaron trabajos de investigación sobre Spark en conferencias académicas, y poco después de su implementación en 2009, ya era diez a veinte veces más rápido que MapReduce para ciertos trabajos. Algunos de los primeros usuarios de Spark fueron otros grupos dentro de UC Berkeley, incluidos investigadores de aprendizaje automático

como el proyecto Mobile Millennium, que utilizó Spark para monitorear y predecir la congestión del tráfico en el área de la Bahía de San Francisco. Sin embargo, en muy poco tiempo, múltiples organizaciones externas comenzaron a usar Spark.

Actualmente hoy más de cincuenta se contabilizan en la página de patrocinio de dicha herramienta de software y docenas hablan sobre sus casos de uso en eventos como: Spark Meetups y Spark Summit. Además de UC Berkeley, los principales contribuyentes a Spark incluyen Databricks, Yahoo e Intel.

En 2011, el AMPLab comenzó a desarrollar componentes de nivel superior en Spark, como Shark (Hive on Spark) y Spark Streaming. Estas tecnologías a veces se denominan Berkeley Data Analytics Stack (BDAS). Spark se abrió por primera vez en marzo de 2010 y fue transferido a la Fundación Apache Software en junio de 2013, donde ahora es un proyecto de alto nivel.

Spark

Es una plataforma de computación en clúster, diseñada para ser rápida y de uso general. Por el lado de la velocidad, Spark amplía el popular modelo MapReduce, para soportar de manera eficiente más tipos de cálculos; incluidas consultas interactivas y procesamiento de flujo [70]. La velocidad es importante en el procesamiento de grandes conjuntos de datos, ya que significa la diferencia entre explorar de forma interactiva y esperar minutos u horas.

Una de las características principales que Spark ofrece para la velocidad es la capacidad de ejecutar cálculos en la memoria, de manera más eficiente que MapReduce en aplicaciones complejas que se ejecutan en disco. Por el lado de la generalidad, Spark está diseñado para cubrir una amplia gama de cargas de trabajo, que anteriormente requerían sistemas distribuidos separados, incluidas aplicaciones por lotes, algoritmos iterativos, consultas interactivas y transmisión. Al admitir estas cargas de trabajo en el mismo motor, Spark hace que sea fácil y económico combinar diferentes tipos de procesamiento,

lo que a menudo es necesario en los flujos de análisis de datos de producción.

Además, reduce la carga administrativa de mantener herramientas separadas. Spark está diseñado para ser altamente accesible, ofreciendo API en Python, Java, Scala y SQL, y bibliotecas agregadas. También se integra estrechamente con otras herramientas de Big Data. En particular, Spark puede ejecutarse en clústeres de Hadoop y acceder a cualquier fuente de datos de este último, incluida Cassandra [70].

Pila de tecnologías Spark

El proyecto Spark según [70]: contiene múltiples componentes estrechamente integrados. En esencia, Spark es un "motor computacional" que es responsable de programar, distribuir y monitorear aplicaciones que consisten en múltiples tareas computacionales distribuidas en máquinas de trabajo, o un grupo de cómputo. Debido a que el motor central de Spark es rápido y de uso general, alimenta múltiples componentes de alto nivel especializados para diversas cargas de trabajo, como SQL o aprendizaje automático. Estos componentes están diseñados para interactuar estrechamente, lo que le permite combinarlos como bibliotecas en un proyecto de software.

Una filosofía de estrecha integración tiene varios beneficios. Primero, todas las bibliotecas y componentes de nivel superior en la pila se benefician de las mejoras en las capas inferiores. Por ejemplo, cuando el motor central de Spark agrega una optimización, las bibliotecas SQL y de aprendizaje automático también se aceleran automáticamente. En segundo lugar, los costos asociados con la ejecución de la pila se minimizan, porque en lugar de ejecutar varios sistemas de software independientes, una organización necesita ejecutar solo uno. Estos costos incluyen implementación, mantenimiento, pruebas, soporte y otros. Esto también significa que cada vez que se agrega un nuevo componente a la pila de Spark, cada organización que use Spark podrá probar este nuevo componente de inmediato. Esto cambia el costo de probar un nuevo tipo de análisis de datos desde la descarga, implementación y aprendizaje de un nuevo proyecto de software hasta la actualización de Spark.

Finalmente, una de las mayores ventajas de una integración estrecha es la capacidad de programar aplicaciones que combinan a la perfección diferentes modelos de procesamiento. Por ejemplo, en Spark puede escribir una aplicación que utilice el aprendizaje automático para clasificar los datos en tiempo real a medida que se ingieren de fuentes de transmisión [70]. Simultáneamente, los analistas pueden consultar los datos resultantes, también en tiempo real, a través de SQL, por ejemplo, para unir los datos con archivos de registro no estructurados. Además, los científicos de datos más sofisticados pueden acceder a los mismos a través del shell Python. Otros pueden acceder a través de aplicaciones por lotes independientes. Todo el tiempo, el equipo de TI tiene que mantener un solo sistema. En la Figura 2.26 presentaremos brevemente cada uno de los componentes de un clúster Spark. La parte inferior de la ilustración se compone de YARN, MESOS o despachadores independientes, estos se encargan de gestionar recursos y servicios, en la parte intermedia se encuentra Spark, tecnología que como ya se mencionó, permite el acceso a gran velocidad de los datos. Finalmente, la parte superior de la ilustración se conforma de librerías que permiten acceso a los algoritmos de *machine learning* como MLib; visualización de grafos (Graphx); acceso a datos con consulta avanzadas SQL (Spark SQL).

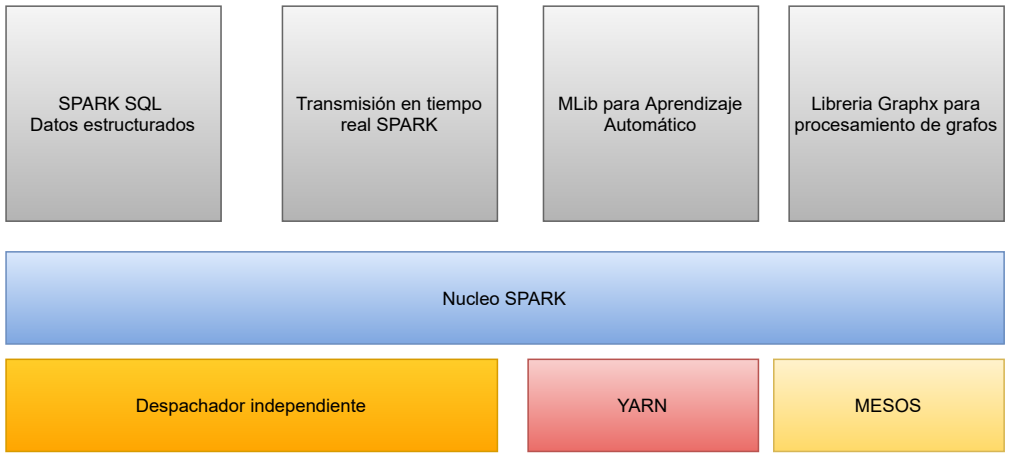


Figura 2.26: Representación gráfica de stack SPARK.

Núcleo SPARK

Contiene la funcionalidad básica de Spark, que incluye según [70]: componentes para la programación de tareas, administración de memoria, recuperación de fallas, interacción con sistemas de almacenamiento y más. Spark Core también alberga la API que define conjuntos de datos distribuidos resistentes (RDD), que son la abstracción de programación principal de Spark. Los RDD representan una colección de elementos distribuidos en múltiples nodos de cómputo que pueden manipularse en paralelo. Spark Core proporciona diversas API para crear y manipular estas colecciones.

Spark SQL

Es el paquete de Spark para trabajar con datos estructurados. Permite consultas a través de SQL, así como la variante Apache Hive de SQL, llamada Hive Query Language (HQL), y admite variadas fuentes de datos, incluidas tablas Hive, Parquet y JSON. Spark SQL permite a los desarrolladores mezclar consultas SQL con las manipulaciones programáticas de datos compatibles con los RDD en Python, Java y Scala, todo dentro de una sola aplicación, combinando así SQL con análisis complejos. Esta estrecha integración con el rico entorno informático proporcionado por Spark hace que este paquete sea diferente a cualquier otra herramienta de almacenamiento de datos de código abierto [70].

Adquisición de datos con Spark

Es un componente de Spark [70], que permite el procesar datos transmitidos en tiempo real. Por ejemplo, archivos de registro generados por servidores web, de producción o colas de mensajes con actualizaciones de estado publicadas por los usuarios de un servicio web. El API para manipular flujos de datos que coinciden estrechamente con la API RDD (para conjuntos de datos resilentes) de Spark, facilita a los programadores desarrollar un proyecto y agilizar la forma en que manipulan datos almacenados en la memoria, en el disco o que llegan en tiempo real. Debajo de su API, Spark Streaming fue diseñado para proporcionar el mismo grado de tolerancia a fallas, rendimiento y

escalabilidad que Spark Core [70].

Spark en modo de clúster

Apache Spark cuenta un administrador de clúster incorporado. Pero puede ejecutarse en Hadoop YARN, Amazon EC2 y Apache Mesos [70].

Arquitectura de tiempo de ejecución

Antes de ejecutar Spark en un clúster, es importante comprender la arquitectura distribuida de este. Como se muestra en la Figura 2.27, este se basa en una arquitectura maestro-esclavo. El maestro se llama conductor y los esclavos se llaman ejecutores [70]. Aun y cuando Spark funciona en una sola máquina, hay una arquitectura distribuida (un controlador con varios ejecutores). El controlador lanza su propio proceso Java, y cada ejecutor corre en un proceso separado del lenguaje mencionado [70].

El conjunto de controladores y ejecutores se conoce como una aplicación Spark. La de esta última es siempre la misma, en clúster de esta herramienta. Además, cada máquina física tiene su propio servicio de administrador. Si tiene más de una máquina, se inicializa utilizando el ejecutor [70].

Controlador SPARK

Es el proceso donde se ejecuta SparkContext. Está a cargo de implementar y ejecutar transformaciones y acciones en RDD. Cuando ejecuta el comando de *shell* Spark en su computadora portátil, en realidad está implementando un programa de controlador. Su primera tarea es llamar el SparkContext, denominado SC (acrónimo de Spark Context). Cuando el programa del controlador termina la aplicación procede de la misma manera. Un programa de usuario básicamente aplica transformaciones y acciones en uno o más RDD, generando nuevos conjuntos de datos, utilizados para cálculos o almacenamiento de los mismos. Otra función del controlador Spark es generar un gráfico acíclico dirigido (DAG, por sus siglas en inglés) de una operación. Con el mencionado, el controlador sabe qué tareas están asignadas a qué nodo, así que si perdió uno de ellos, el controlador sabe

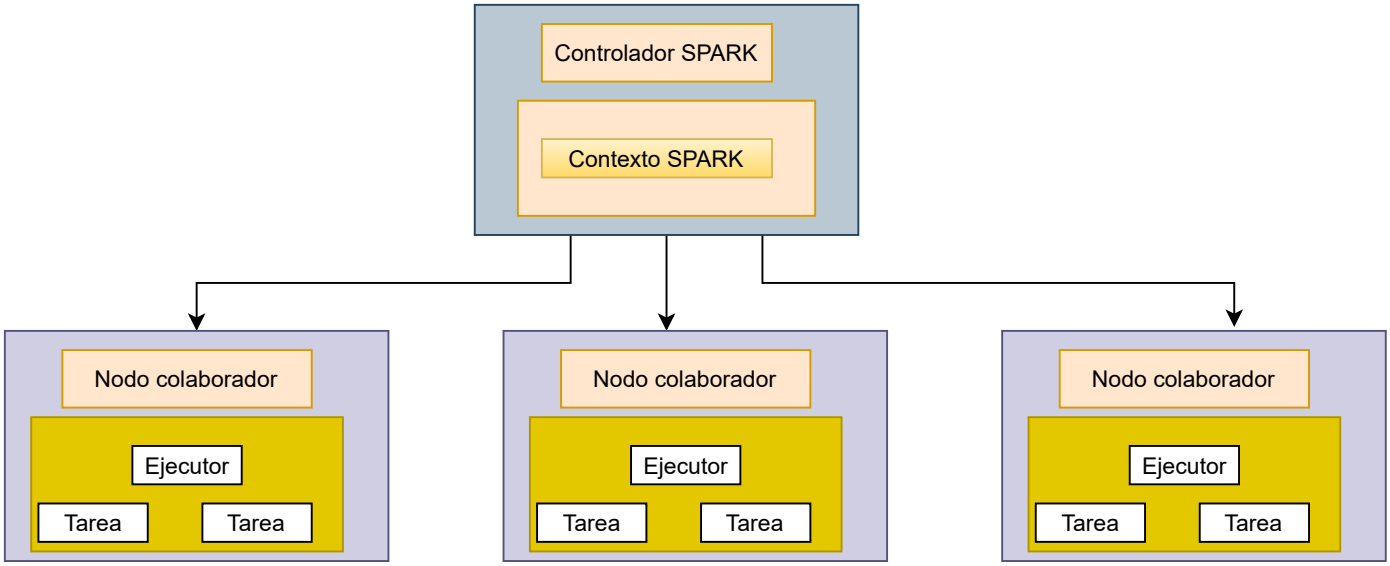


Figura 2.27: Mecanismo de funcionamiento Spark.

en qué punto se encontraba y reasignarlas. El controlador también realiza optimizaciones de canalización, divide el DAG en etapas, cada una con múltiples tareas. En Spark, estas ultimas se definen como las unidades de trabajo más pequeñas. Un programa normal puede lanzar miles de tareas [70].

Programación de tareas

Dado un plan de ejecución física, el controlador Spark coordina las tareas que realiza cada nodo ejecutor. Cuando este comienza a funcionar, se registra en el controlador, por lo que este último siempre tiene una vista completa de todos los nodos del ejecutor. Cada uno de los mencionados, es un proceso Java independiente que puede ejecutar tareas y almacenar RDD. Cuando se lanza un programa, el controlador lo subdivide en tareas, monitorea los nodos ejecutores disponibles e intenta equilibrar la carga de trabajo entre ellos. Además, también sabe qué parte de los datos tiene cada nodo para reconstruir todo al final. El controlador muestra su información en la Web, para que el usuario siempre pueda ver lo que está sucediendo, de forma predeterminada, se ejecuta

en el puerto 4040. Los ejecutores tienen dos roles según [70]:

- Ejecutar las tareas asignadas y entregar resultados al controlador.
- Proporcionar almacenamiento en memoria para RDD. Un programa llamado Block Manager se ejecuta en cada ejecutor y administra la memoria y los RDD. Al ejecutar Spark en modo local, el controlador y los ejecutores se ejecutan en el mismo proceso. Esto es para fines de desarrollo; no se recomienda en un entorno de producción.

Clúster Manager Spark

La herramienta Spark, actualmente proporciona todos los elementos necesarios para implementar un clúster de Big Data, sin recurrir a otras herramientas de terceros, antes había que integrar Spark con otros proveedores de administración de clústers. Integrado con la distribución Spark se le conoce como administrador independiente, pero es un componente que puede cambiarse y usar un administrador personalizado como Hadoop Yarn, Amazon EC2 o Apache Mesos. Es importante tener en cuenta que los términos controlador y ejecutor se usan cuando se habla de la aplicación Spark. Cuando hablamos del administrador de clúster, usamos los términos maestro y esclavo. Es importante no confundir los términos o intercambiarlos, porque son conceptos diferentes. Independientemente del administrador de clúster utilizado, Spark proporciona un solo *script*, denominado *spark-submit*, para arrancar la ejecución de las tareas de Spark. Además puede conectarse a diferentes administradores a través de varias opciones y gestionar los recursos del clúster que la aplicación necesita [70].

2.2.4. Componentes de lectura y escritura de datos

Esta clase de componentes está orientado a la escritura y recuperación de datos usando el modelo tradicional de un lenguaje de consultas estructurado. De ese modo, el programador acostumbrado a operar bases de datos tradicionales no percibe de manera tan contundente

el cambio de paradigma hacia un modelo donde la variedad en los datos está presente. A continuación se describen algunos de estos componentes.

Hive

Es un componente de software que permite almacenamiento y consultas de manera estructurada. De esa manera, para especialistas de bases de datos, acostumbrados a los modelos SQL estándar, la migración hacia el uso de *Big Data* puede ser más atractiva con este tipo de herramientas, puesto que, desde la perspectiva del desarrollo no representa un cambio. Sin embargo, detrás de este componente de software existe un sistema *Big Data*, con sistemas y almacenamiento distribuido, además, replicaciones de los datos [71].

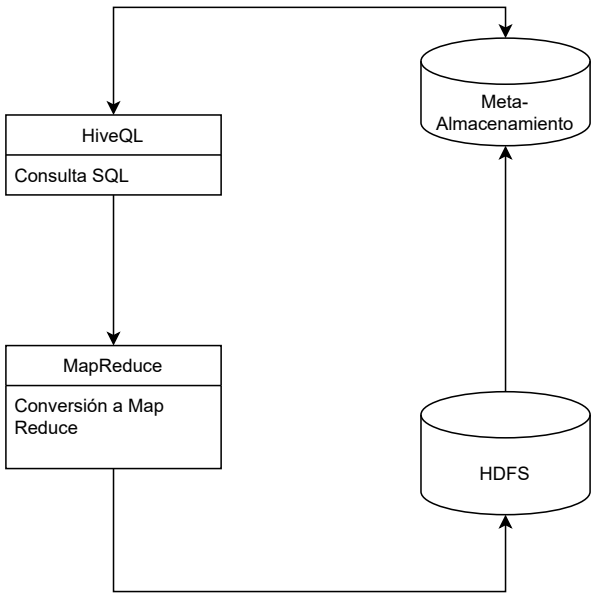


Figura 2.28: Proceso de funcionamiento del componente Hive

Accumulo

Es un componente parecido a Hive, descrito en el párrafo anterior, ofrece un manejo sencillo de acceso a los datos con un alto desempeño. Diseñado para aplicaciones de

redes de datos, representadas por grafos, tales como: análisis de documentos, registros de salud, bioinformática, redes sociales [72].

HBase

Es una solución NoSQL, que permite almacenar los datos usando un mecanismo de clave-valor. Además, fue de las primeras soluciones implementadas incluso antes que Hive [73]. En la Figura 2.29, tomada de [66], muestra el esquema de proceso de Hbase. Un coordinador externo (Zookeeper) controla el servicio de Hbase, a su vez que, un maestro gestiona servidores denominados de región, estos se encargan de mantener un segmento de información de un volumen masivo almacenado en el sistema de archivos de Hadoop.

Hawq

Es un motor de procesamiento masivo SQL, similar a Hive, pero con mayor velocidad y con la capacidad de ejecutarse en memoria principal [74].

HiveSQL

Es el componente que permite acceder a los datos usando el lenguaje de consultas estructuradas estándar SQL. Se puede decir que es el complemento de la solución de almacenamiento Hive [75].

Pig

Este componente provisto de un motor para ejecución de flujos de datos en paralelo sobre Apache Hadoop. Además, Incluye un lenguaje llamado Pig Latin, mismo que utiliza operadores tradicionales de SQL, Por ejemplo: unión, ordenamiento y filtrado, por mencionar algunos. Por consiguiente, los usuarios de esta herramienta se pueden familiarizar al seguir aplicando esquemas como los tradicionales de motores de bases de datos relacionales [76].

SparkSQL

Es un componente que permite a los programadores de Spark aprovechar los benefi-

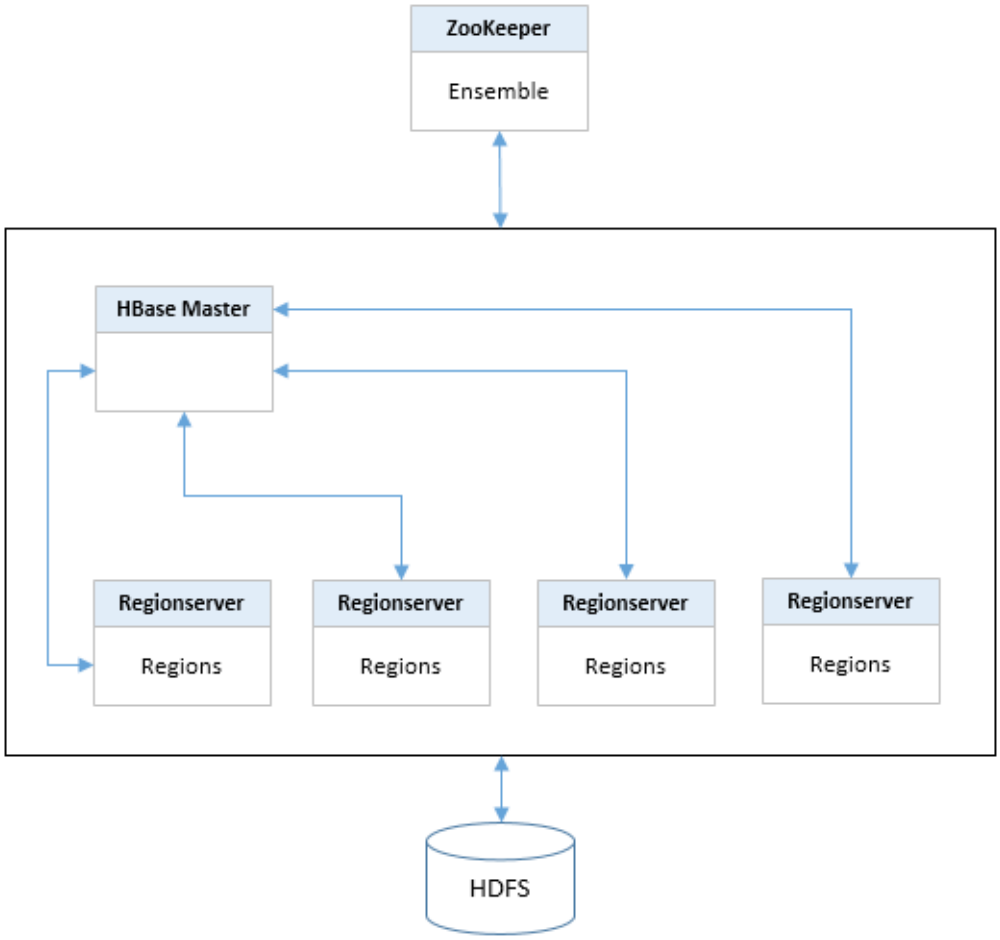


Figura 2.29: Esquema de proceso de trabajo Hbase

cios del procesamiento relacional. Consultas declarativas, almacenamiento optimizado y conexión con bibliotecas de Spark son algunas de sus ventajas [77].

2.2.5. Componentes de adquisición en lote y por flujo continuo

Estos componentes de *Big Data* se utilizan según la necesidad de acceso a los datos, es decir, en tiempo real o por lotes. Algunos datos han sido generados por sistemas de software y guardados en almacenes. De ese modo, son procesados en lote. Sin embargo, aplicaciones donde se requiere adquirir datos en tiempo real necesitan otro tipo de componentes para realizar dicha acción. Entre ello podemos mencionar:

Sqoop

Es un componente permite la adquisición de datos usando fuentes variadas, tales como: bases de datos relacionales externas y almacenarlos en el sistema de archivos Hadoop (HDFS, por sus siglas en inglés), para su procesamiento. Además, Sqoop es compatible para la transferencia de datos entre HDFS y sistemas de bases de datos relacionales, por ejemplo: MySQL y Oracle. Este proceso se ilustra en la Figura 2.30, tomada y modificada de [66], donde el gestor relacional a través de sqoop intercambia datos con el sistema de archivos Hadoop.

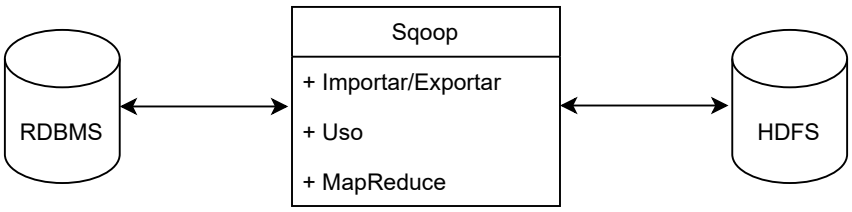


Figura 2.30: Funcionalidad Sqoop

Flume

Es un *framework* distribuido que gestiona la adquisición y agregación de datos en tiempo

real. Especialmente datos de tipo evento y *bitácora*. En la Figura 2.31, se puede apreciar el proceso de adquisición, transporte y almacenamiento de datos usando Flume.

Figura 2.31: Funcionalidad de Flume

Storm

Es un componente para escribir aplicaciones que procesen rápidamente cantidades masivas de datos en tiempo real de forma distribuida en un modelo de cómputo paralelo y con tolerancia a fallas [64].

Kafka

Es un orquestador para registrar el orden en que llegan los datos cuando son adquiridos por medio de *Streaming*.

2.2.6. Componentes para análisis y visualización de datos

Los lenguajes de programación son un componente clave para realizar el análisis de cantidades masivas de datos. De tal manera, gran parte de las librerías de *Machine Learning* son accesibles por medio de un entorno de programación. A continuación se describen algunos ellos utilizados por la capacidad y eficiencia mostrada en aplicaciones *Big Data*, para múltiples industrias. Entre ellos, diversos libros de trabajo digitales para proporcionar visualizaciones en las plataformas Spark y Hadoop. Utilizando códigos de programación y en complemento con librerías gráficas se pueden visualizar datos y ordenarlos bajo criterios estadísticos básicos, tales como: valor máximo, mínimo, promedio, desviación estándar, Un ejemplo se puede observar en la Figura 2.33. Algunas de las herramientas más importantes

son el IPython Notebook (Jupyter), Spark Notebook, Ispark, Hue, Spark Kernel, Jove Notebook, Beaker Notebook y Databricks Cloud. Todos estos cuadernos son de código abierto, excepto Databricks Cloud [55].

Scala

Según [38], es un lenguaje de programación que se utiliza para construir aplicaciones de minería de datos. Abarca las eficiencias de desarrollo de Java y se enfoca en el paradigma de programación funcional, todo con una sintaxis más concisa. Scala, fue diseñado para para crear aplicaciones grandes, paralelas y distribuidas. Comercialmente es utilizado por firmas de diversar industrias, tales como: FourSquare, LinkedIn, The Guardian, Novell, Siemens, Sony y Twitter, entre otros.

Python

Según [78], es un lenguaje de programación interpretado, interactivo, y orientado a objetos. Actualmente es compatible con entornos como Spark y es popularmente conocido como el lenguaje de ciencia de datos. Este lenguaje integra estructuras de datos de alto nivel tales como: listas y arreglos asociativos (conocidos como diccionarios), tipificación y unión dinámica, módulos, clases, módulos, excep-ciones, administración automática de memoria, por mencionar algunas. La sintaxis de este lengua-je es simple y elegante, sin dejar de lado el enfoque de un lenguaje de propósito general. Diseñado en 1990 por Guido van Rossum; este lenguaje es gratuito aun para aplicaciones comerciales y es compatible con cualquier computadora moderna en sistemas como Linux, Windows y Mac.

R

Nació como un proyecto para ser utilizado en entorno académico, ha evolucionado al grado de obtener reconocimiento en una extendida y gran comunidad de ciencia de datos. Con diferentes aplicaciones en entornos académicos y una creciente aceptación en el sector privado. Este lenguaje es tan robusto que tiene paquetes para funciones que

van desde el análisis del habla, la ciencia genómica, la minería de texto, entre otros [38].

Flujos de datos con Zepellin

Los componentes de la arquitectura Zeppelin se describen de la siguiente manera [55]:

- *Frontend*: Es la interfaz de usuario para mostrar datos en forma tabular, gráfica y exportar información. Se ilustra en la Figura 2.32, tomada directamente del clúster implementado en este trabajo de tesis.
- Zeppelin Server: Es la parte que permite ejecutar trabajos por medio de sockets web y la API DE REST para interactuar con la interfaz de usuario y el servicio de acceso de forma remota.
- Sistema de intérprete instantaneo: Se refiere a que se pueden activar de forma manual los interpretes requeridos al momento. De ese modo, el usuario puede interactuar con ellos activandolos por medio de un menu. Intérpretes como Spark, Shell, Markdown, AngularJS, Hive, Ignite, Flink y otros, son algunos ejemplos.
- Intérpretes: Cada intérprete se ejecuta en una JVM separada para proporcionar la funcionalidad que necesita el usuario.

Jupyter

Jupyter Notebook, se puede definir como libros interactivos para visualización de datos a través de una interfaz web. Compatible con leguajes tal como: Python, Scala, R y Julia. Tiene opciones para configurar el idioma. Jupyter proporciona la siguientes características [55]:

1. Un *shell* interactivo para comandos del sistema operativo
2. Una consola Qt para análisis interactivos basados en shell
3. Análisis interactivos en un navegador web

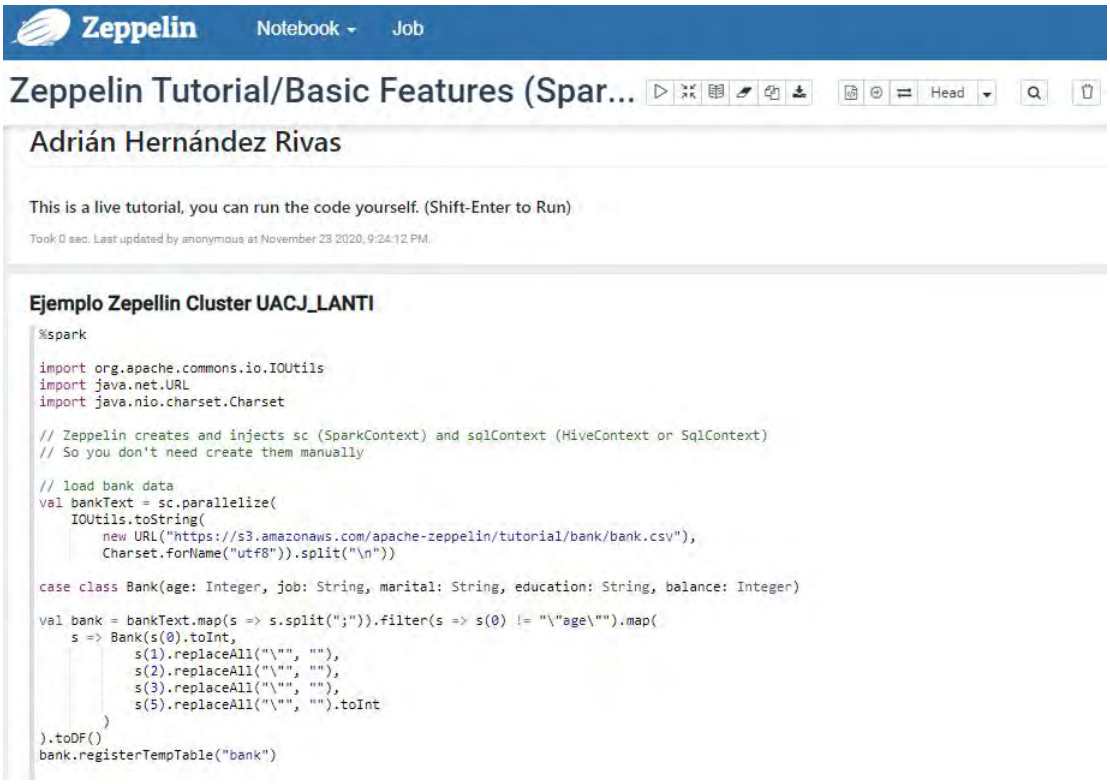


Figura 2.32: Componente Zepellin

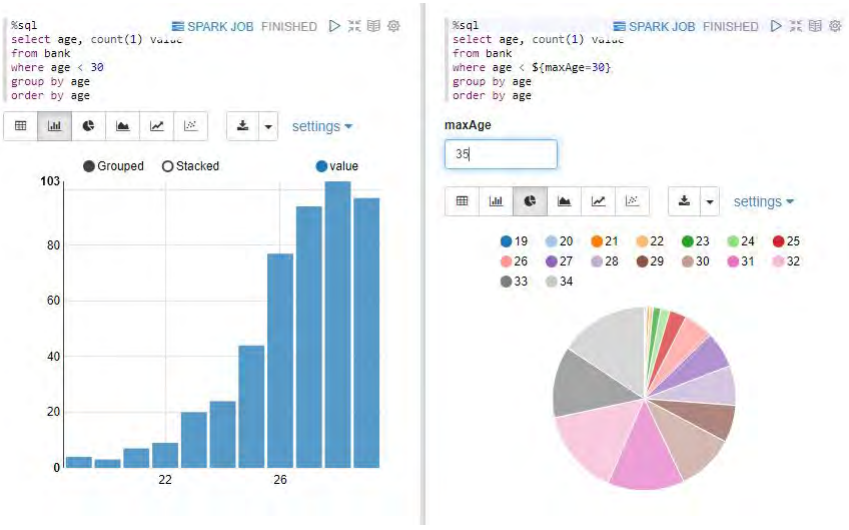


Figura 2.33: Visualización de datos en Zepellin

Aplicacion	Herramientas	Técnicas
Transformación de datos y enriquecimiento	MapReduce: procesamiento de Hadoop marco de referencia Spark: motor de cálculo Hive: almacenamiento de datos y consultas Pig: lenguaje de flujo de datos Python: programación Crunch, Cascading, Scalding y Cascalog: Herramientas especiales de MapReduce	Munging de datos Filtración Unión ETL Formato de archivo con Anonimización Re-identificación
Analítica de datos	Hive: almacenamiento de datos y consultas Pig: lenguaje de flujo de datos Tez: alternativa a MapReduce Impala: alternativa a MapReduce Ejercicio: alternativa a MapReduce Apache Storm: motor de cálculo en tiempo real Spark Core: motor de cálculo Spark Core Spark Streaming: motor de cálculo en tiempo real Spark SQL: para análisis de SQL SolR: plataforma de búsqueda Apache Zeppelin: cuaderno basado en web Cuadernos Jupyter Nube de Databricks Apache NiFi: flujo de datos Conector Spark-on-HBase Lenguajes de programación: Java, Scala y Phyton	Analítica en línea Procesamiento analítico en línea Procesamiento de datos Visualización de datos Procesamiento de eventos complejo Procesamiento y transmisión en tiempo real Búsqueda de texto completo Analítica de datos interactivos
Ciencia de datos	Python: programación R: lenguaje de computación estadística Mahout: biblioteca de aprendizaje automático de Hadoop MLlib: biblioteca de aprendizaje automático de Spark GraphX y GraphFrames: marco de procesamiento gráfico de Spark y datagramas de adopción de gráficos.	Analítica predictiva Análisis de sentimiento Texto y natural Procesamiento de lenguaje Análisis de red Análisis de clústeres

Tabla 2.4: Herramientas Big Data.

4. La herramienta nbconvert para convertir .ipynb un libro interactivo de esta herramienta a otros formatos como .html, .pdf, markdown, entre otros.

5. La herramienta nbviewer (http://nbviewer.ipython.org/) permite ver los libros interactivos por medio de internet con la integración de GitHub para compartir cuadernos en dicho repositorio de código.

Criterios de selecciόm de componentes para plataformas *Big Data*

La selección de componentes de software que pueden integrarse a una plataforma como servicio para *Big Data* puede variar y justificarse en base a los requerimientos de la aplicación que se implementará, por el tipo de datos con los que cuenta la organización y que requiere analizar en las búsqueda de anomalías, en áreas tales como: finanzas, servicios, ventas, dispositivos, por mencionar algunos.

Los componentes presentados en este Capítulo 2, en las subsecciones: 2.2.4, 2.2.5, 2.2.6,

tienen prestaciones que pueden ser complejas de comprender para un usuario sin conocimientos técnicos en profundidad del tema *Big Data*. De tal modo, se presenta al lector una guía basada en preguntas enfocadas a la parte operativa en relación al negocio. en la que las respuestas pueden asociarse de con uno de los componentes presentados.

- ¿Qué tipo de datos serán almacenados? basándose en la pregunta: ¿Cuál es mi información de partida y como está estructurada?
- ¿Qué tipo de informes o visualizaciones se van a generar?, ¿Estadísticos, grafos conectados, formularios dinámicos?
- ¿Cuáles herramientas son requeridas en la construcción de modelos predictivos basándose en el cómputo que requieren dichos modelos?
- ¿Los datos se van adquirir y analizar en tiempo real?
- ¿Los datos se van adquirir por flujo continuo o por lote?
- ¿Existe información previa y cómo está estructurada?
- ¿Qué infraestructura tengo disponible local o de nube?
- ¿Qué tipo de tecnologías dominan los involucrados y con cuáles están familiarizados?

Es decir según las condiciones y requerimientos del negocio, existe uno o varios componentes que se adapta para tales efectos. Por ejemplo, si una empresa necesita adquirir datos de sensores y analizarlos en tiempo real para monitoreo de fallas, existen componentes específicos que se pueden seleccionar para dichos fines, tal como: *kafka*, *Apache spark*, *Elasticsearch* o *Grafana*, por mencionar algunos. En la Tabla 2.4, tomada de [55], se clasifican los componentes de software que pueden dar soporte a las distintas actividades que se llevan a cabo en un proceso de analítica de datos.

Capítulo 3

Implementación del clúster

En el presente capítulo se describen las actividades realizadas relacionadas con el objetivo general: diseño e implementación de una plataforma *Big Data* para la detección de anomalías.

3.1. Descripción de la arquitectura del clúster

A fin de llevar a cabo una implementación de la plataforma como servicio para *Big Data* ha sido necesario definir una arquitectura que incluya la especificación de los recursos de software y hardware necesarios para el despliegue y adecuada operación del proyecto. En caso contrario, una limitación de infraestructura puede resultar en un proyecto incompleto e inoperable.

Arquitectura lógica

La arquitectura lógica permite identificar los componentes, servicios e interacción dentro de la arquitectura implementada en el clúster de *Big Data*.

Sistema operativo de clúster

En la Figura3.1, se ilustra la manera en que DCOS como sistema operativo de clúster es la base de toda la implementación de la plataforma como servicio y sus componentes *Big*

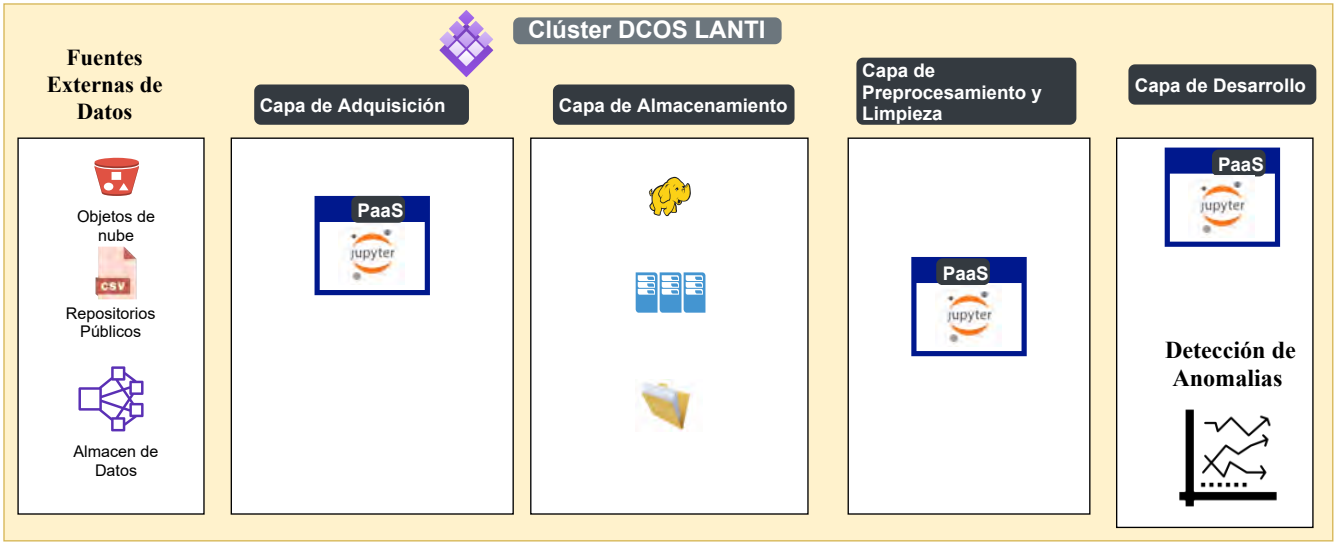


Figura 3.1: Arquitectura Lógica.

Data, la elección de este sistema operativo de clúster se basa en los siguientes criterios:

- Capacidad de integrar la mayoría de los *Frameworks* de procesamiento y analítica de datos como: Spark, Mesos, Mahout, Hadoop, Mahout, Kafka, Pig, Hive, Flume.
- El esquema de distribución de tareas y trabajos operan orquestados por el *kernel Mesos*, esto garantiza un esquema de cómputo distribuido. Por consiguiente, otra de las ventajas es la capacidad de escalar el número de equipos de cómputo pertenecientes al clúster desplegado en DCOS.
- Capacidad de comunicación interna entre los *frameworks*. Es una ventaja importante para los procesos de *Big Data* involucrados que necesitan compartir datos e información entre ellos, cuando esto no sucede el tiempo y complejidad que representan las tareas de coordinación y configuración requeridos son altos. Por el contrario, DCOS se encarga de dicha tarea, sin dejar de lado que su modelo de plataforma como servicio es otro factor que contribuye en reducir las dificultades en este aspecto. Adicionalmente, para implementación de un clúster *Big Data*, [12] recomienda tener en cuenta lo siguiente:

Construir un sistema escalable y eficiente que soporte una amplia gama de *frameworks* de analítica de datos actuales y futuros

El clúster DCOS que se implementó en este trabajo permite la escalabilidad horizontal. De ese modo, si los recursos computacionales se agotan, una nueva integración de nodos está permitido siguiendo las instrucciones de configuración[61].

Fuentes externas de datos

El origen de datos en este experimento se centró en fuentes de datos de objetos de nube y conjuntos de repositorios públicos en internet. Sin embargo, el clúster también tiene la capacidad de adquirir datos de otras fuentes, tales como: almacén de datos, sensores, redes sociales, por mencionar algunos.

Capa de adquisición de datos

La primer actividad, como se ilustra en la Figura 3.1, consiste en la adquisición de datos en crudo. Por consiguiente, para esta tarea se diseñó una arquitectura capaz de acceder a los diversos formatos de información provenientes de fuentes externas, tales como: bases de datos, objetos de S3 de Amazon, archivos separados por coma (disponibles en repositorios públicos en internet y también de manera local desde un almacenamiento en red o en disco). Tal como se ilustra en la Figura 3.1, los orígenes de datos pueden provenir de más de una origen externo. Por consiguiente, existe más de un método de acceder a los datos, por ejemplo: API, ODBC, conexiones a un NFS, clúster Hadoop o de bases de datos NoSQL.

Para acceder a la información guardada se utiliza la lógica de intercambio de información entre aplicaciones *Middleware* (por su significado en inglés), utilizando rutinas de código según que proporcionan los lenguajes de programación desde el componente Jupyter Labs agregado a la implementación.

Capa de almacenamiento

Para esta actividad lo recomendable es centralizar la información en una colección de datos organizada según las prioridades del negocio y que será accedida posteriormente

para su procesamiento y análisis. Para dicho fin, se utilizan lagos de datos que son almacenes para la información. En este trabajo se implementó un clúster Hadoop para tales efectos. Adicionalmente, se pueden guardar algunos conjuntos de datos dentro del entorno de trabajo de JupyterLabs, disco local o de red. En la Figura 3.1, la capa de almacenamiento se ilustran las opciones para guardar la información, tales como NFS, clúster Hadoop y entorno de desarrollo Jupyter. Sin embargo, para sistemas *Big Data*. Es recomendable utilizar Hadoop si no es necesario un análisis en tiempo real y se requiere una alta tasa de respuesta en la lectura y escritura de altos volúmenes de datos.

Hadoop, según [67], es un marco de trabajo que permite procesamiento distribuido de cantidades masivas de información usando mecanismos de programación básica. Entre las características por las que fue seleccionada esta herramienta de Big Data se mencionan las siguientes [70]:

- Procesamiento distribuido, esto significa que los datos se pueden procesar en varios servidores.
- Eficiente, esta característica se presenta siempre y cuando el volumen de los archivos inicia en orden del Terabyte, no es recomendado usar archivos de pequeña cantidad de memoria.
- Económico, es una herramienta que se liberó bajo código abierto y licencia de uso libre, por lo que su costo de licenciamiento actual es nulo.
- Escalable, se puede aumentar la arquitectura de hardware sobre la que está desplegada esta herramienta.
- Tolerante a fallos, permite recuperación en caso de corrupción de archivos o falla de algún servidor.
- Open Source, comunidades de desarrolladores trabajan en la mejora de esta herramienta.

- Capacidad de procesar volúmenes grandes de datos como: terabytes, petabytes, por mencionar algunos.
- Previsión de crecimiento a futuro, si los datos a ser almacenados están en constante crecimiento se puede implementar un fácil escalamiento estableciendo el aumento de recursos manualmente.
- Tipos de datos variados: texto, imágenes, archivos comprimidos, entre otros se pueden manipular con Hadoop.
- Procesamiento paralelo: es la capacidad de particionar nuestros datos, lo que trae como beneficio acelerar la velocidad con que se manipulan estos últimos.

Capa de preprocesamiento y limpieza

Una vez que se tiene un repositorio de almacenamiento de datos, los mismos se pueden acceder para el preprocesamiento que abarca tareas de visualización inicial, limpieza y normalización, según sea requerido. Para tales efectos, se sigue utilizando el repositorio Hadoop para acceder la información a través de las herramientas de JupyterLabs. Sin embargo, este último tiene limitaciones que son resueltas en gran medida por las capacidades de Spark, entre ellas se mencionan las siguientes:

- Permite trabajar con datos en memoria principal en lugar de disco, para evitar la saturación de la misma, se usa la paralelización descrita en el siguiente punto. Además, esta característica permite acelerar iteraciones durante ejecución de los algoritmos de aprendizaje automático.
- La paralelización de los datos es directamente proporcional a la cantidad de CPU disponibles de Spark. Por ejemplo, un conjunto de cien datos se puede dividir en cuatro subconjuntos de veinticinco. Esto sucede porque se tienen asignados para la herramienta cuatro unidades de procesamiento.
- Spark permite el procesamiento de datos por lotes y por adquisición directa de los sensores.

En la Figura 3.1, este proceso es ilustrado a través del componente Jupyter. Este último es accesible desde el PaaS para realizar dicho proceso.

Capa de desarrollo

Este proceso consiste en construir modelos de aprendizaje automático con el fin de encontrar patrones escondidos dentro de la información relacionados con la detección de anomalías. De ese modo, para contribuir positivamente con el negocio u organización los componentes para dicho fin en esta arquitectura son: Librerías de aprendizaje automático del lenguaje Python, de visualización gráfica como Matplot y otros *frameworks* como SparkML. Además, dentro de esta categoría de componentes el entorno de desarrollo JupyterLabs integra otros lenguajes de ciencia de datos como R y Scala que también son herramientas alternativas para la tarea de construcción de modelos. En la Figura 3.1 estos componentes están disponibles a través del Paas y el componente JupyterLabs.

Visualización de datos

El paso final de esta arquitectura consiste en presentar la información en libros de notas interactivos proporcionados por el entorno JupyterLabs. En la Figura 3.1, este último es considerado el entorno de desarrollo.

Descripción de arquitectura física

A continuación se describen los elementos y cantidad que fue requerida para integrar la arquitectura final del clúster, ver Figura 3.2, tales como: memoria en disco, principal , procesador y máquinas virtuales con la capacidad suficiente para desplegar y operar todos los componentes que conforman la plataforma como servicio para Big Data y detección de anomalías.

Unidades de procesamiento

Se asignaron 38 unidades de procesamiento central o CPU (por siglas en inglés) para el clúster. Las herramientas de software actualmente instaladas tienen un consumo cercano a diez CPU según información arrojada por el monitor de recursos del clúster. Es

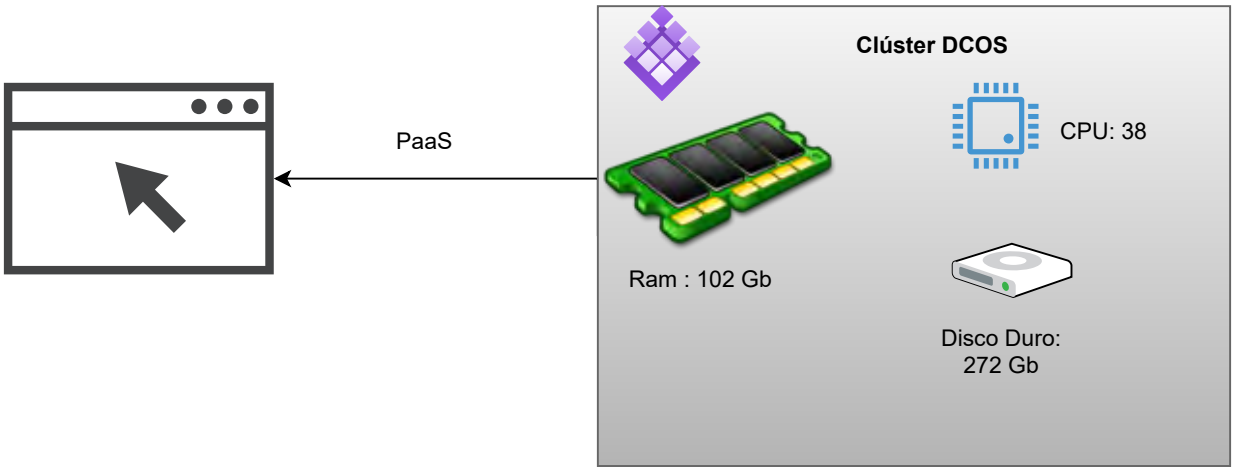


Figura 3.2: Hardware del clúster.

importante mencionar que las herramientas de software pueden utilizar fracciones de procesador, por lo tanto, estos últimos se aprovechan de forma más eficiente.

Memoria de acceso aleatorio

La memoria de acceso aleatorio RAM (por sus siglas en inglés) utilizada es de 102 Gigabytes. Actualmente existe un uso exclusivo por las aplicaciones instaladas en el clúster aproximado de 27.3 GB.

Almacenamiento en disco

Un total de 274 Gigabytes de almacenamiento en disco se asignó a la implementación del clúster.

Monitor de recursos del clúster DCOS

Una vez instalado DCOS en el clúster, al ingresar al mismo, el primer elemento desplegado es un monitor de recursos de hardware (ver Figura 3.3). Entre los elementos que podemos apreciar se encuentran el uso de procesadores, memoria y disco, por mencionar solo algunos.

Arquitectura por nodos

La Figura 3.5 ilustra los equipos de cómputo (nodos), que integran el clúster, así como su

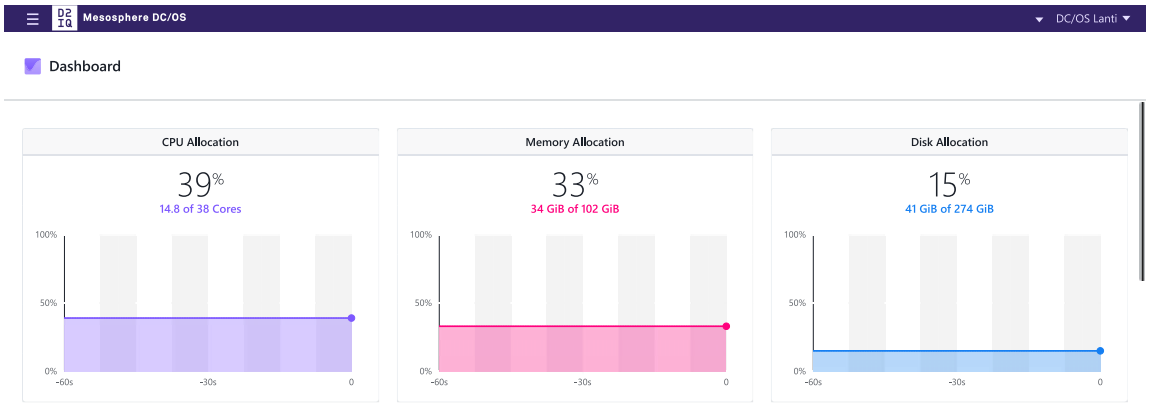


Figura 3.3: Monitor de recursos del clúster DCOS LANTI.

arquitectura de hardware. Estos recursos se suman como si formaran una supercomputadora. En la Figura 3.4, se muestra una ilustración general de los nodos incorporados en la implementación del clúster, agrupados por su funcionamiento, existen los siguientes nodos: maestro, *Bootstrap*, agentes públicos y privados. Los primeros se encargan de asignar y repartir las tareas sobre los agentes; el segundo, tiene por objetivo lanzar actualizaciones y paquetes de software entre los nodos; los últimos, son los encargados de ejecutar las tareas asignadas.

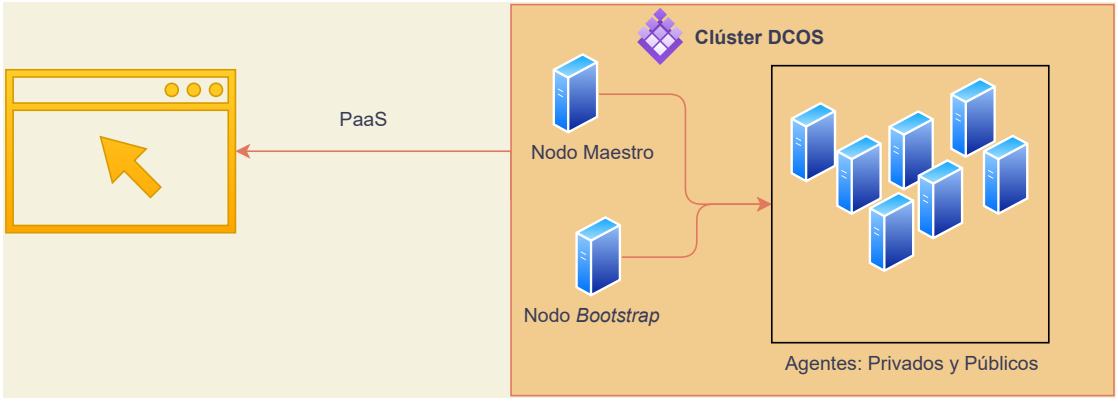


Figura 3.4: Nodos del clúster DCOS LANTI.

En la Tabla 3.1, se muestra cada nodo que es equivalente a una máquina virtual y la

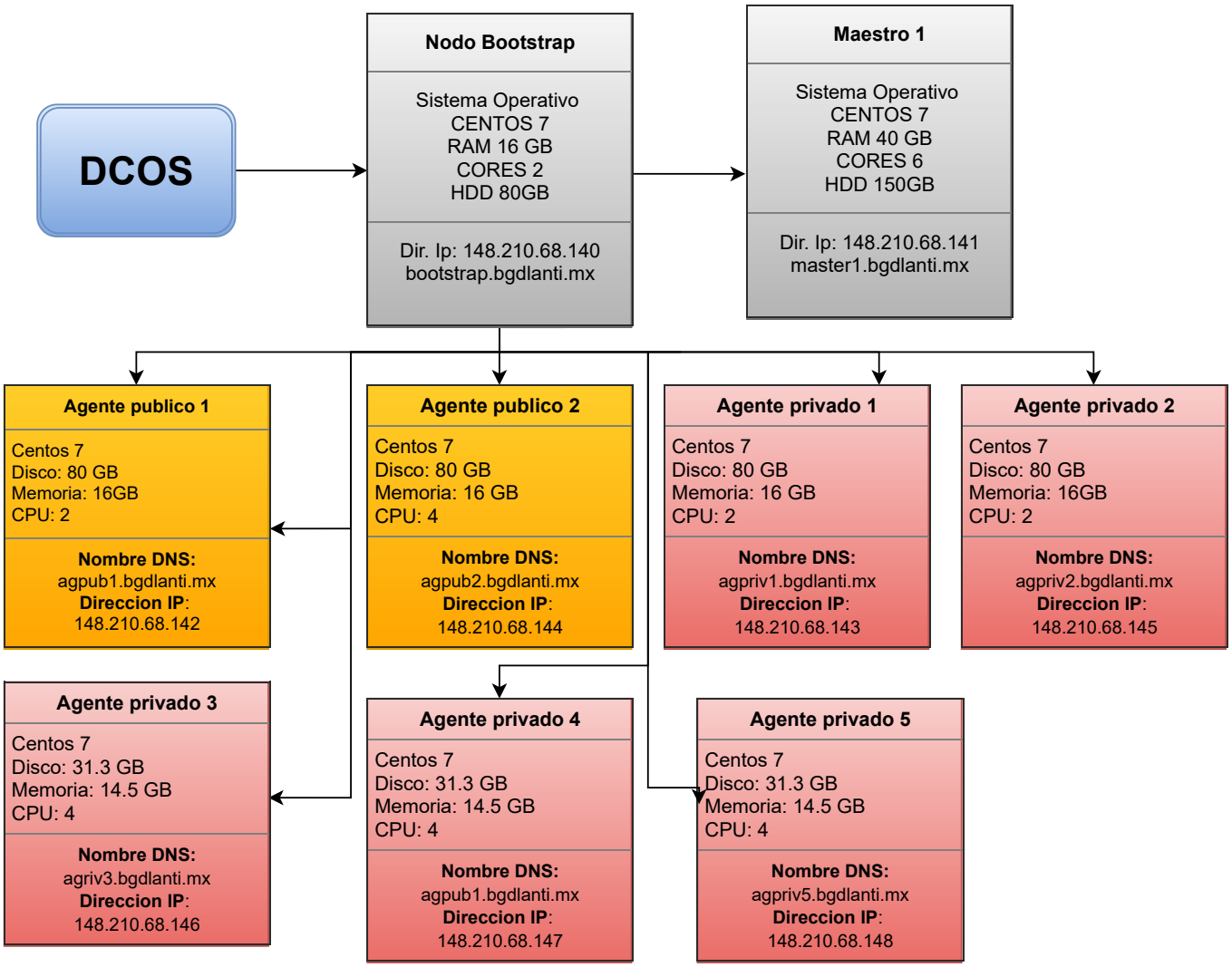


Figura 3.5: Arquitectura de clúster LANTI.

Nombre MV	Descripción	Capacidades
master_node	Máquina virtual con CentOs que realiza la función del nodo maestro del clúster.	S.O.:CentOs 7.7 RAM: 40GB HDD: 150GB CPU: 6
bootstrap_node	Máquina virtual con CentOs que realiza la función del nodo bootstrap para lanzar instalaciones o actualizaciones.	S.O.:CentOs 7.7 RAM: 16GB HDD: 80 GB CPU: 2
agentpub1_node	Máquina virtual con CentOs que realiza la función del nodo agente público donde correrán las aplicaciones.	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2
agentpub2_node	Máquina virtual con CentOs que realiza la función del nodo agente público donde correrán las aplicaciones.	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2
agentpriv1_node	Máquina virtual con CentOs que realiza la función del nodo agente privado donde correrán las aplicaciones	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2
Agentpriv2_node	Máquina virtual con CentOs que realiza la función del nodo agente privado donde correrán las aplicaciones	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2
Agentpriv3_node	Máquina virtual con CentOs que realiza la función del nodo agente privado donde correrán las aplicaciones	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2
Agentpriv4_node	Máquina virtual con CentOs que realiza la función del nodo agente privado donde correrán las aplicaciones	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2
Agentpriv5_node	Máquina virtual con CentOs que realiza la función del nodo agente privado donde correrán las aplicaciones	S.O.:CentOs 7.7 RAM: 16 GB HDD: 80 GB CPU: 2

Tabla 3.1: Hardware utilizado para cluster DCOS.

cantidad de recursos que consumen.

3.2. Preparación de los nodos del clúster

En esta sección se presentan las configuraciones requeridas por cada nodo incorporado al clúster. Por ejemplo, cada uno debe tener instalado un sistema operativo propio [61]. Para esta implementación se instaló Linux en su distribución CentOs 7.7 con los servicios básicos de un sistema operativo servidor en red. Además, cabe mencionar que, se debe utilizar la misma versión para todos los nodos del clúster. La Figura 3.6 ilustra el conjunto de configuraciones requeridas en forma general, tal como: Sistema Operativo, configuración de interfaz de red y nombre de dominio para cada nodo incorporado al clúster. A continuación se enlistan las actividades generales básicas de configuración por nodo, tal como:

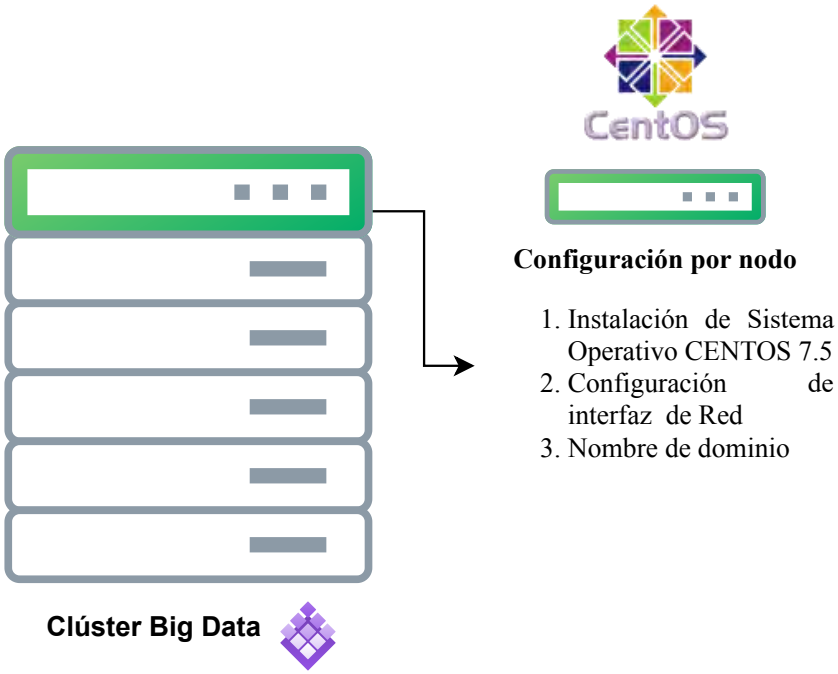


Figura 3.6: Arquitectura por nodo.

1. Configuración de servicios

El requisito para la integración de un nodo al clúster es haber cumplido con una preconfiguración general sobre el mismo que consiste en instalar librerías, habilitar servicios y ejecutar comandos básicos y validar las siguiente características [61]:

- Sistema Operativo. CentOs 7.7, todos los tipos de nodos destinados a formar parte del clúster DCOS (*bootstrap*, maestros y agentes) deben tener el mismo sistema operativo.
- XFS. Es un sistema de archivos de alto rendimiento para el registro de bitácoras que es configurado durante el proceso de instalación de CentOs.
- Se habilitó IPv4 (Protocolo de red en su cuarta versión de 32 bits) y se asignó a todos los nodos una IP (Internet Protocol, por sus siglas en inglés), este último es un protocolo de comunicaciones para dispositivos y computadoras conectadas dentro de una red.
- La zona horaria se configuró con NTP (Network Time Protocol, por sus siglas en inglés).
- A cada nodo del clúster, se le asignó un nombre de dominio, usando el sufijo: bgdlanti.mx y prefijo el tipo de nodo por ejemplo: master1.

2. Configuración de puertos y protocolos

Además de las configuraciones anteriores, también es necesario activar las siguientes características que activan la comunicación interna entre nodos, como lo indica [61]:

- Secure Shell (SSH, por sus siglas en inglés), debe estar habilitado en todos los nodos. Este es un protocolo basado en el algoritmo de seguridad informática RSA (Rivest Shamir Adleman, por sus siglas en inglés); por lo tanto, es necesario activarlo para poder conectarse de forma remota a cada nodo.
- El protocolo de mensajes de control de internet (ICMP, por sus siglas en inglés), debe estar habilitado en todos los nodos.
- Cada nodo tiene acceso a la red desde el nodo *bootstrap*, en otras palabras deben tener comunicación interna a través de la red.

- Cada nodo tiene conectividad a todos los demás nodos y viceversa, sin restricciones en el clúster DCOS.
- El protocolo UDP (User Datagram Protocol, por sus siglas en inglés), el cual permite el intercambio de datagramas sin establecer una conexión previa, debe estar abierto para ingresar al puerto 53. Este puerto se utiliza para escanear los nodos maestros en el clúster ejecutando: `find leader.mesos`.

3. Registro de maestros y esclavos

- El comando que permite activar el servicio de registro de nodo maestro o esclavo se debe ejecutar con la siguiente linea de código: `mesos agent node service`
- Para activar un nodo en modo esclavo se utilizó el siguiente comando: `dcos-mesos-slave`.

Parametrización de nodos

Para la instalación adecuada del sistema operativo de clúster, es necesario la identificación y preconfiguración de algunos paquetes y librerías en los nodos maestros, esclavos y *bootstrap*. Para dicha tarea se ejecutan una serie de comandos y scripts que establecen el rol de funcionamiento de cada nodo en el clúster. [61].

3.3. Preparación, instalación y despliegue DCOS

En esta sección se describen las actividades necesarias para realizar una instalación del clúster con DCOS. Para semiautomatizar esta tarea actualmente se utiliza un estándar para descarga de paquetes con estructura de archivo similar al lenguaje extensible de marcado XML (Xtensible Markup Language, por su siglas en inglés) y denominado YAML (YAML Ain't Markup Language, siglas del acrónimo recursivo en inglés). En la Figura 3.7, se ilustra el proceso planteado en este párrafo, pasar parámetros a un archivo YAML (1) y (2); configuración de *bootstrap*(3); inyección DCOS en los nodos agentes (4) y Despliegue por medio de una interfaz web al PaaS en DCOS (5).

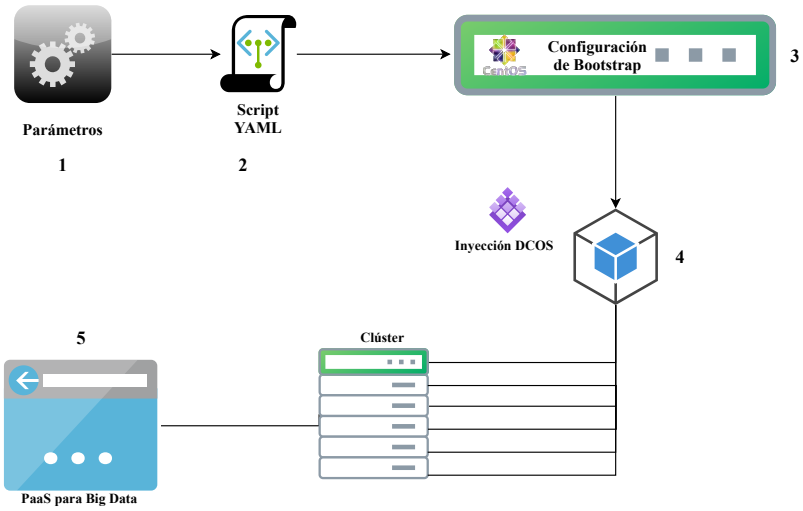


Figura 3.7: Instalación DCOS.

1. Archivo de configuración YAML

Es un formato de serialización de datos legible por humanos, sirve para generar los archivos de instalación del clúster. Se recomienda guardarlo como: genconf/config.yaml

```
bootstrap_url: http://148.210.68.140:80
cluster_name: DC/OS Lanti
exhibitor_storage_backend: static
master_discovery: static
ip_detect_public_filename: /genconf/ip-detect
master_list:
- 148.210.68.141
resolvers:
- 8.8.8.8
- 8.8.4.4
ssh_key_path: genconf/ssh_key
ssh_port: 22
ssh_user: root
```

2. Agregar nombres de hosts

Es importante agregar en el archivo *hosts* del directorio `/etc` el nombre de dominio y la IP de cada uno de los nodos que integran el clúster. Esta acción es necesaria para la instalación de algunas herramientas de software, en un proceso posterior a la instalación del clúster DCOS. El servidor de nombres dominio (DNS, por sus siglas en inglés) sirve para dicha tarea en un entorno de producción, en esta implementación no se cuenta con uno. Por tal motivo se añadieron de forma manual las direcciones de red. Las siguientes lineas de comandos sirven para añadir nodos y nombres de dominio:

```
127.0.0.1
localhost localhost.localdomain
localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain
localhost6 localhost6.localdomain6
148.210.68.145  agpriv2.bgdlanti.mx agpriv2
148.210.68.141  master1.bgdlanti.mx master1
```

3. Instalación DCOS

A partir de estos pasos inicia la preparación de *scripts* y configuraciones relacionadas con la instalación DCOS.

4. Descarga de archivo de configuración

(`dcos_generate_config`), se debe descargar en el nodo *bootstrap* y es utilizado para generar un *script* de compilación, necesario para arrancar el proceso de instalación DCOS. El siguiente comando `curl` es utilizado para la tarea mencionada, el comando completo se muestra a continuación:

```
curl -O https://downloads.dcos.io/dcos/stable/dcos_generate_config.sh
```

5. Generación del archivo de compilación

Es necesario ingresar al nodo *bootstrap* y ejecutar el *script* de *shell* para generar un archivo de compilación personalizado del instalador DCOS. En este proceso se extrae

un contenedor Docker con archivos genéricos y que pueden ser personalizados para una configuración avanzada del clúster. Estos últimos, son enviados al directorio `./genconf/serve/`. Las opciones del instalador son visibles en una terminal con la línea siguiente de código: `dcos_generate_config.sh --help`

comando para generar archivo de compilación: `sudo bash dcos_generate_config.sh`

```
| --dcos-genconf.<HASH>.tar
  |-----dcos_generate_config.sh
     |----- genconf
        |--- config.yaml
        |--- ip-detect
```

6. Empaquetado de la instalación DCOS

Como usuario con permisos de administrador se ejecuta el siguiente comando para alojar el paquete de instalación de DCOS a través de un contenedor NGINX Docker. Además, es necesario especificar el valor del puerto que se utiliza en la dirección de red del nodo *bootstrap*.

```
sudo docker run -d -p 8081:80 -v
$PWD/genconf/serve:/usr/share/nginx/html:ro nginx
```

7. Ejecutar el instalador DCOS

Para inicializar el lanzador de instalación de DCOS usando el correspondiente archivo de compilación es necesario ejecutar los siguiente comandos y acciones:

- Acceder a cada nodo maestro, a través del protocolo ssh, ejecutar el siguiente comando: **ssh 148.210.68.141**
- Se define un nuevo directorio y se navega al mismo. `mkdir /tmp/dcos && cd /tmp/dcos`
- Descargue el instalador DCOS del contenedor NGINX Docker, de 148.210.68.140 y puerto 80 se especifican en la URL de arranque.

```
curl -O http://148.210.68.140:80/dcos_install.sh
```

- Ejecute el siguiente comando para instalar DCOS en sus nodos maestros.

  `sudo bash dcos_install.sh master`

- Agentes privados: `sudo bash dcos_install.sh slave`
- Agentes publicos: `sudo bash dcos_install.sh slave_public`

8. Primer acceso al clúster DCOS LANTI

A través de un navegador de internet y usando la red local de la UACJ, se ingresa en la barra de dirección la IP del nodo maestro: http://148.210.68.141/ es la forma de acceder al clúster DCOS, el primer ingreso se realiza por medio de una cuenta de Gmail, Github o Microsoft Windows, en esta instalación utilizamos cuentas de Gmail como cuentas de administrador.

9. Instalación DCOS CLI

Es una utilería en modo de línea de comandos para la instalación de herramientas de software a través de una terminal en el nodo Maestro. *install CLI*, se muestra en la parte superior derecha como se puede ver en la Figura 3.9. El uso de CLI es opcional. Al ingresar al menú mencionado se muestra el siguiente comando:

```
curl https://downloads.dcos.io/cli/releases/binaries/dcos/
linux/x86-64/latest/dcos -o dcos && chmod +x ./dcos &&
sudo mv dcos /usr/local/bin &&
dcos cluster setup http://148.210.68.141 && dcos
```

3.4. Instalación de componentes *Big Data*

En esta sección se describen las actividades realizadas para la instalación de componentes de software descritos en la subsección arquitectura lógica de este mismo Capítulo. Esta

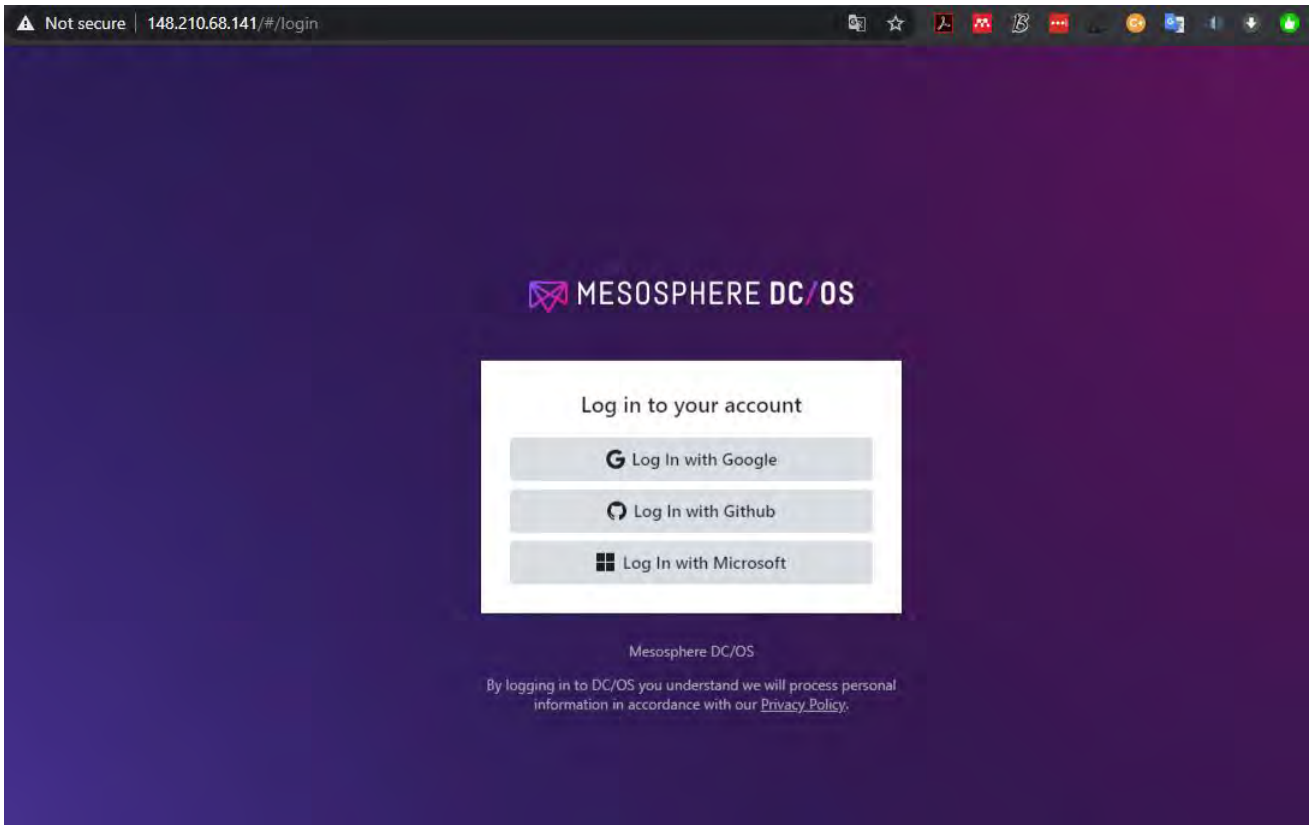


Figura 3.8: Accesso al clúster DCOS.

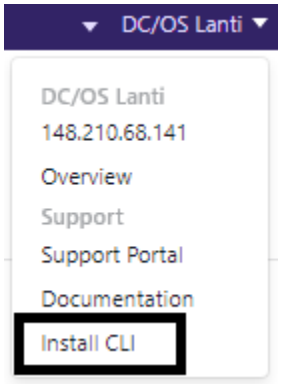


Figura 3.9: Instalación CLI DCOS.

tarea se realizó a través de un asistente gráfico auxiliar en la instalación de paquetes de software (compatibles con Docker). Además, entre los componentes que se describen en la instalación está: Hadoop, diseñado para almacenamiento de grandes volúmenes de datos que pueden ir en el orden del Teratbyte y Petabyte; Spark, para la paralelización y analíticas de datos; DSE (Data Science Engine, por sus siglas en inglés) que es conocido oficialmente como JupyterLabs. Este último permite habilitar un entorno de desarrollo y visualización de la información.

1. Búsqueda para instalar Hadoop a través del asistente gráfico

Localizado en la parte superior izquierda, (ver Figura 3.10), *catalog* permite realizar la búsqueda manual del componente de software requerido mediante el ingreso del nombre, en este caso Hadoop.

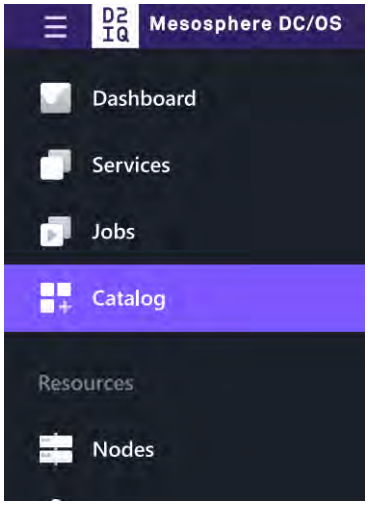


Figura 3.10: Catálogo de software en DCOS.

2. Selección de componente en catálogo

Este proceso se realiza en la interfaz de DCOS; ingresando el nombre del paquete de software en el campo de texto identificado por el icono de una lupa (ver Figura 3.11) y activando la búsqueda con la tecla *Enter*, los resultados son desplegados en tres diferentes pestañas. Como se puede ver en en la Figura 3.11, la primer pestañana (*All*) despliega

los resultados generales de la búsqueda, incluye paquetes validados por comunidades de desarrollo de software (*Certified*) y experimentales (*Community*), que corresponden con la pestaña segunda y tercera mostrada en la Figura 3.11 del mismo título.

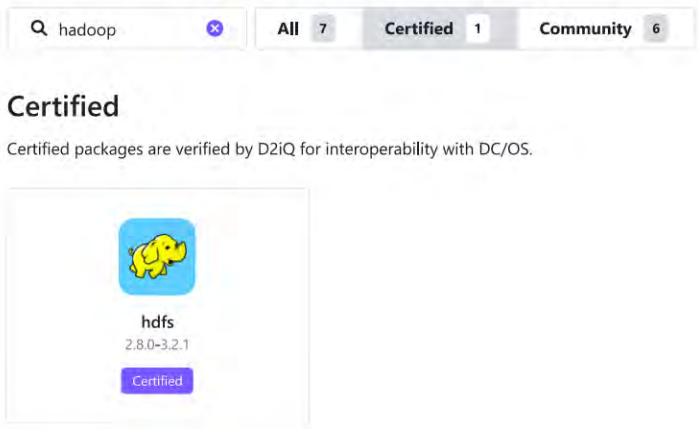


Figura 3.11: Gestor de software DCOS.

3. Revisión de requerimientos

En la interfaz web, mostrada en la Figura 3.12, son desplegados los requerimientos mínimos de hardware solicitados por una una aplicación o componente de software previo a su instalación. *Run Service*, está localizado en la parte superior derecha de Figura 3.12, ejecuta la instalación preconfigurada del software seleccionado.

4. Configuración avanzada de software

Existe la posibilidad de parametrizar Hadoop y en general los paquetes de software a través de un archivo en formato JSON (JavaScript Object Notation, por sus siglas en inglés), en caso de no conocer estos parámetros se deja sin modificación este archivo y se lanza la instalación en el *Review Run*. Este modo de instalación se recomienda para usuarios que conocen las variables y opciones avanzadas de cada componente.

5. Notificación del proceso de instalación

Si el proceso de instalación es llevado a cabo sin ningún tipo de error mostrará un mensaje de *success* como el mostrado en la Figura 3.14.

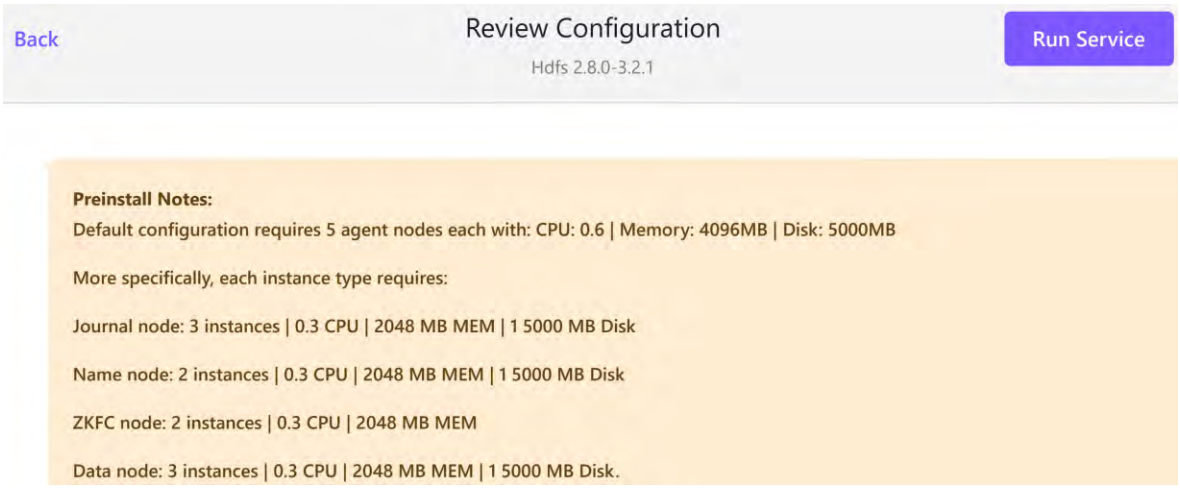


Figura 3.12: Requerimientos previos de Hadoop.

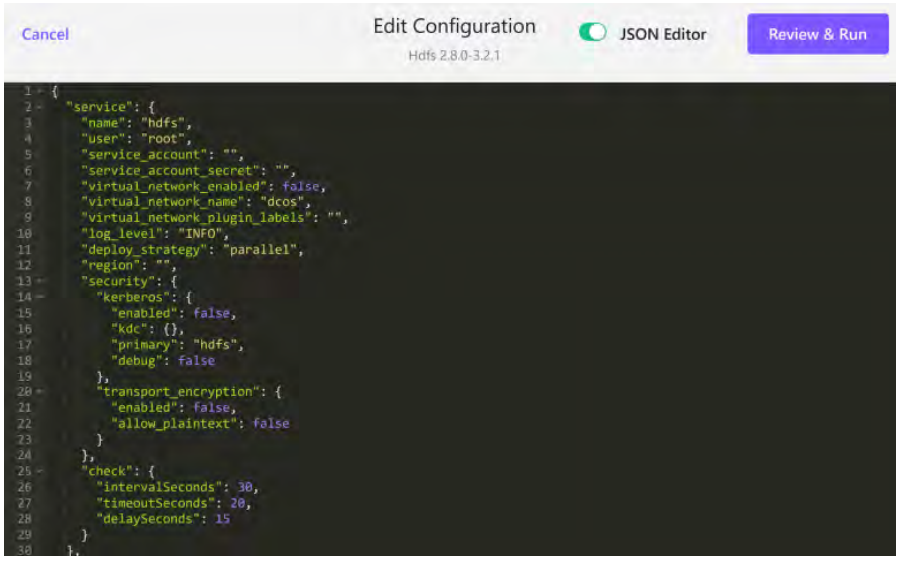


Figura 3.13: Configuración de parámetros en modo JSON.

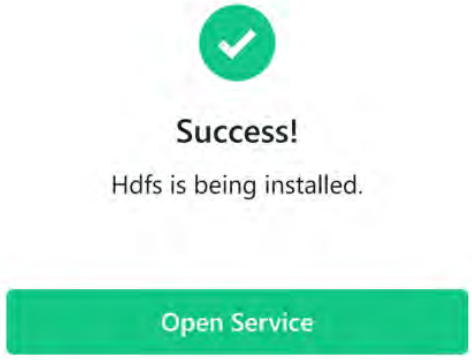


Figura 3.14: Instalación exitosa Hadoop.

6. Verificación del servicio

Para mostrar los servicios en ejecución en el clúster DCOS, acceder a el menú *services* que aparece en la Figura 3.10, Una vez dentro de este menú se muestra la lista de aplicaciones en funcionamiento, ver Figura 3.15.

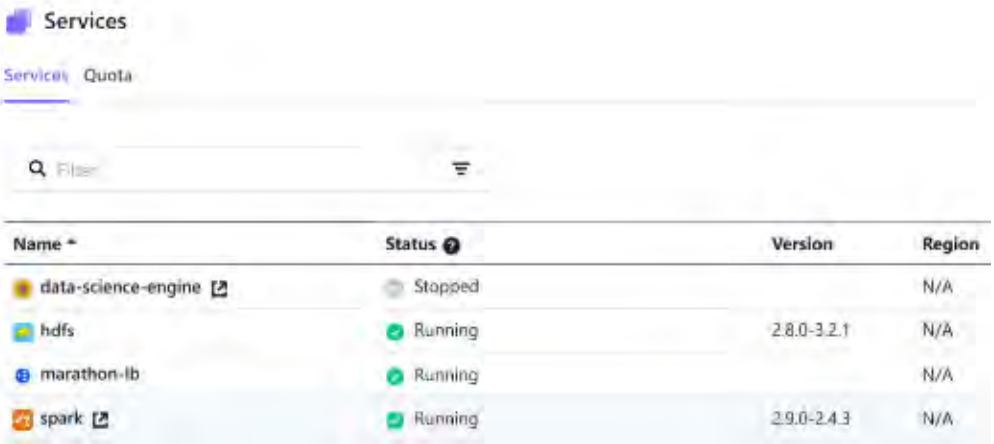


Figura 3.15: Estado de servicios en clúster.

7. Búsqueda del componente Spark

Ingresar al menú *catalog* mostrado en la Figura 3.10, repetir el mecanismo de búsqueda como se realizó con la herramienta Hadoop, ingresar en el campo de texto el término de

Spark.

8. Selección de la versión Spark

De los resultados mostrados en la Figura 3.16, es recomendable instalar versiones certificadas y compatibles con distribuciones Hadoop, para ello es necesario verificar la compatibilidad en las páginas oficiales de dichos componentes.

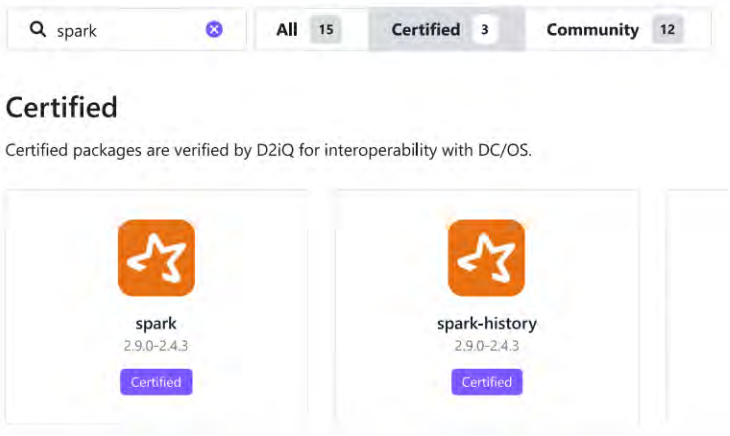


Figura 3.16: Búsqueda de Spark.

9. Verificación de requisitos Spark

Antes del arranque de la instalación Spark, los recursos mínimos de hardware que consume la herramienta son mostrados como se ilustra en la Figura 3.17. Este proceso se repite para cada nuevo componente que será instalado en el clúster.

10. Instalación Spark

Spark se instala usando el mismo procedimiento de Hadoop, con un clic sobre el botón *Review and Run*, mostrado en la Figura 3.18, se ejecuta el proceso encargado de dicha tarea; al terminar correctamente aparece nuevamente el mensaje mostrado en la Figura 3.14. Para verificar el proceso de la nueva herramienta instalada, procedemos a revisar el servicio de la misma forma que accedimos en Hadoop y mostrado en la Figura 3.15.

11. Instalación DSE (JupyterLabs)

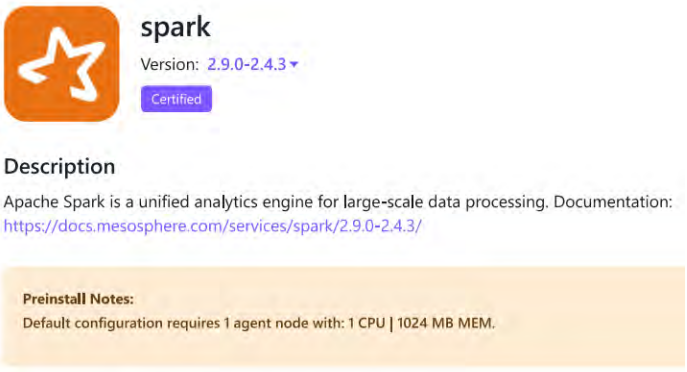


Figura 3.17: Requisitos Spark.

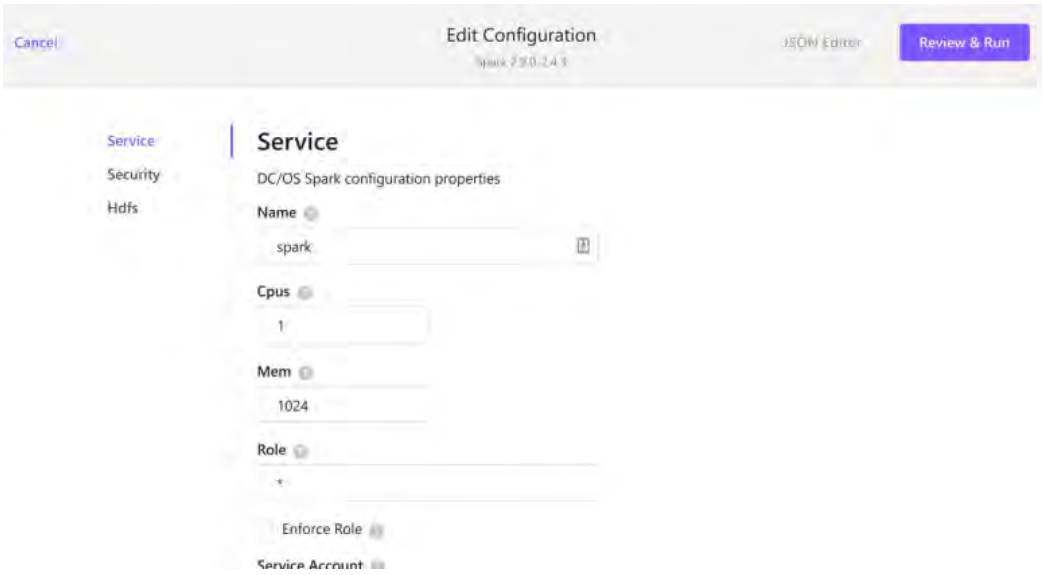


Figura 3.18: Instalación Spark.

Data Science Engine, según describe [61], es un entorno de desarrollo dinámico, permite usar múltiples aplicaciones centralizadas en una *Suite* de software. Entre las cuales se encuentran lenguajes de programación, tales como: Python, R, Scala; conexión con *frameworks* de analítica de datos, Hadoop y Spark; visualización de datos, a través de los conocidos *Notebooks*, definidos como cuadernos web con la siguientes características: se pueden ejecutar segmentos de código de algún lenguaje de programación, visualizar gráficas y mostrar analíticas en tiempo real [55]. Siguiendo el procedimiento de instalación de Hadoop y Spark, de la misma manera se realiza la búsqueda de la herramienta en el catálogo y se lanza la instalación DSE con parámetros predeterminados.

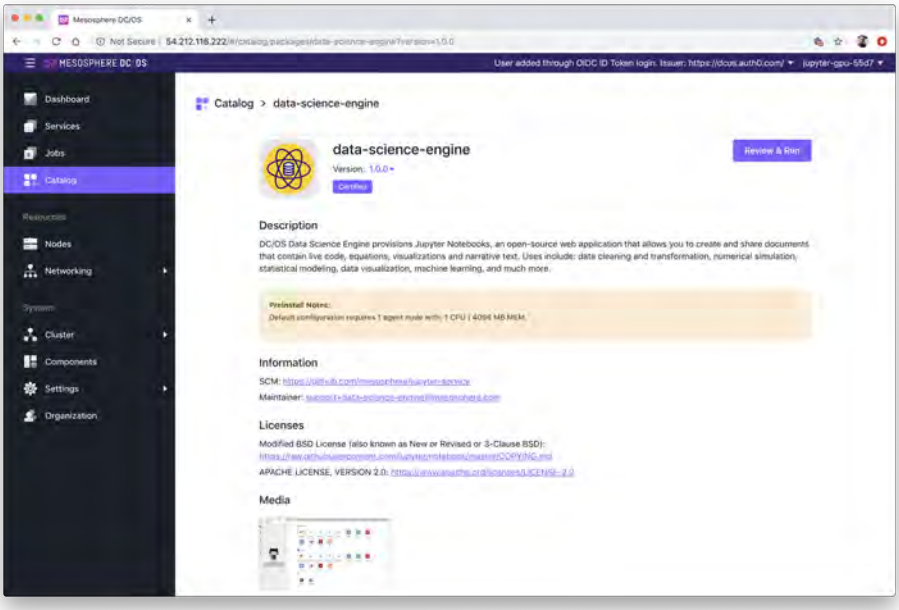


Figura 3.19: Instalación DSE.

3.5. Pruebas de componentes *Big Data*

La realización de pruebas con la implementación tiene como objetivo validar el funcionamiento adecuado de cada uno de los componentes instalados de forma conjunta. Recordando al lector el modelo de arquitectura lógica, el procesamiento de datos se puede llevar a cabo de dos maneras: la primera. consiste en procesar los datos desde un repositorio en la nube y realizar la analítica pertinente (ver Figura 3.20) parte superior. La segunda es adquirir los datos y almacenarlos en Hadoop y despues aplicar un proceso de ciencia de datos sobre los mismos, ver Figura 3.20, parte inferior.

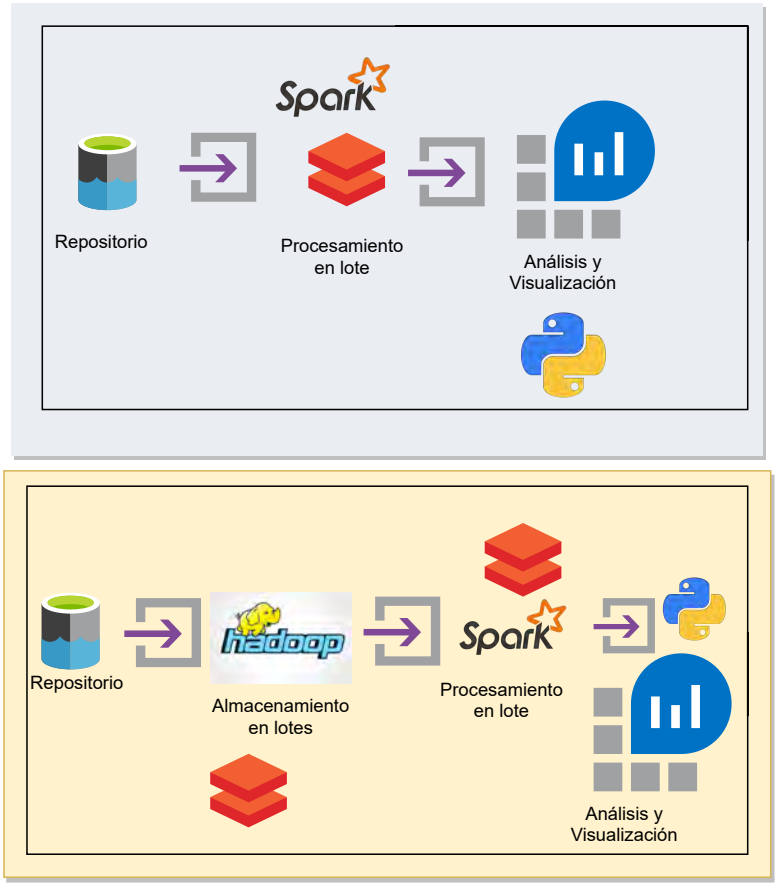


Figura 3.20: Modelos de procesamiento *Big Data*

Además, se describen los procedimientos para integrar las herramientas Hadoop, Spark y entorno de desarrollo DSE para ser utilizadas de manera conjunta y cumplir el siguiente objetivo específico: validar las herramientas de software en cada parte del proceso de analítica de datos: adquisición, almacenamiento, limpieza, preprocesamiento, análisis y visualización.

1. Integración de Hadoop y Spark en Data Sciense Engine

Para configurar las herramientas Hadoop y Spark con DSE, se requiere iniciar sesión en el clúster e ingresar la dirección url del mismo http://148.210.68.141 en la barra de un navegador web y dentro de la red local de la UACJ o externamente usando un cliente vpn con las respectivas credenciales de usuario. Localizar y acceder al menú *services* del clúster DCOS LANTI, ubicar e ingresar al servicio hdfs que muestra dos archivos ubicados en la pestaña endpoints correspondientes a la configuración de Hadoop `hdfs-site.xml` y `core-site.xml`, ver Figura 3.21, con clic en el menú *Download* es necesario descargar al directorio predeterminado del equipo de cómputo del usuario.

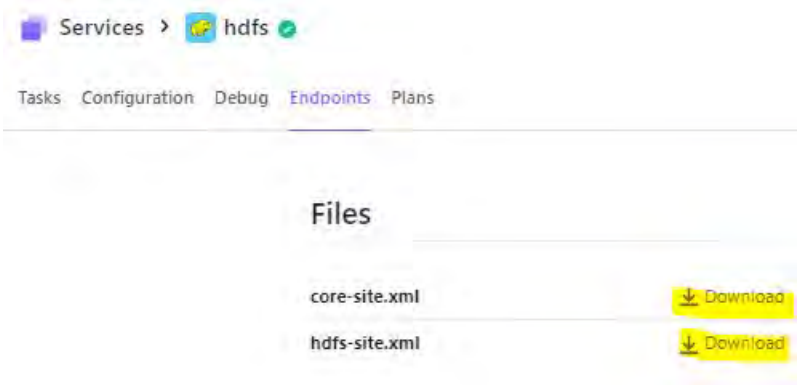


Figura 3.21: Archivos de configuración Hadoop.

2. Acceso a Data Science Engine

En el menú servicios de DCOS ubicar `data-science-engine` y hacer clic sobre el icono mostrado en la Figura 3.22 resaltado en amarillo. En la Figura 3.23, la interfaz solicita una contraseña, por defecto es: jupyter. Al acceder se muestra una interfaz como la de

la Figura 3.24, el directorio remarcado en amarillo se debe acceder con doble clic para subir los archivos de configuración de hadoop, descargados como se mencionó en los pasos previos.

Figura 3.22: Acceso DSE.

Figura 3.23: Servicio DSE.

La interfaz DSE facilita el proceso para subir archivos e integrar configuraciones, con un clic sobre la flecha mostrada en la Figura 3.24 y marcada en amarillo, abre una ventana que permite navegar al directorio de ubicación donde se descargaron los archivos de configuración de Hadoop: `core-site.xml y hdfs-site.xml`, seleccionamos y subimos dichos archivos al folder `hadoop_conf` precargado con DSE.

3. Pruebas de Hadoop

A continuación se describen los pasos para la validación de las herramientas Hadoop y Spark en Data Science Engine. Para el proceso de pruebas con Hadoop, es necesario iniciar una consola desde la interfaz de usuario en DSE localizando el ícono con el símbolo:

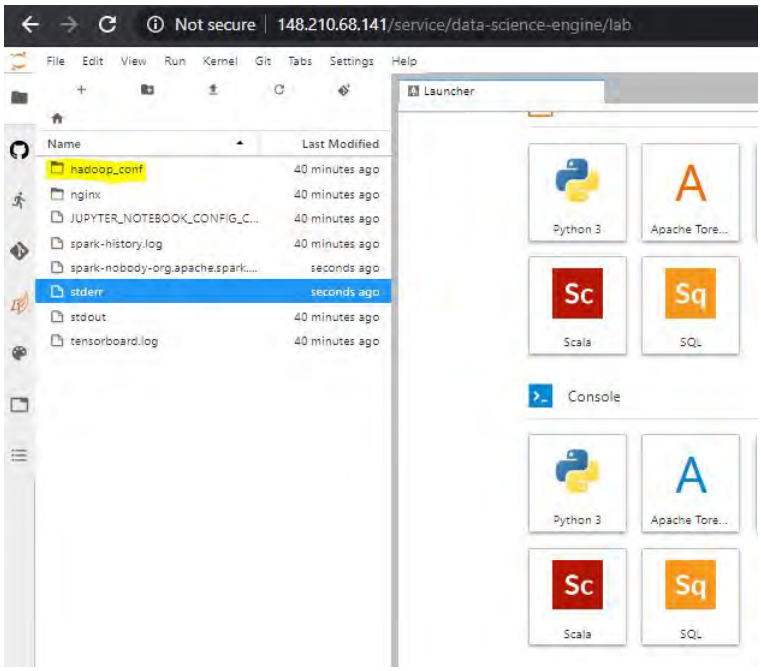


Figura 3.24: Jupyter Notebooks.

Figura 3.25: Ícono para subir archivos.

$_ del conjunto de paquetes ubicado en la parte inferior izquierda y que puede ver en la Figura 3.26.

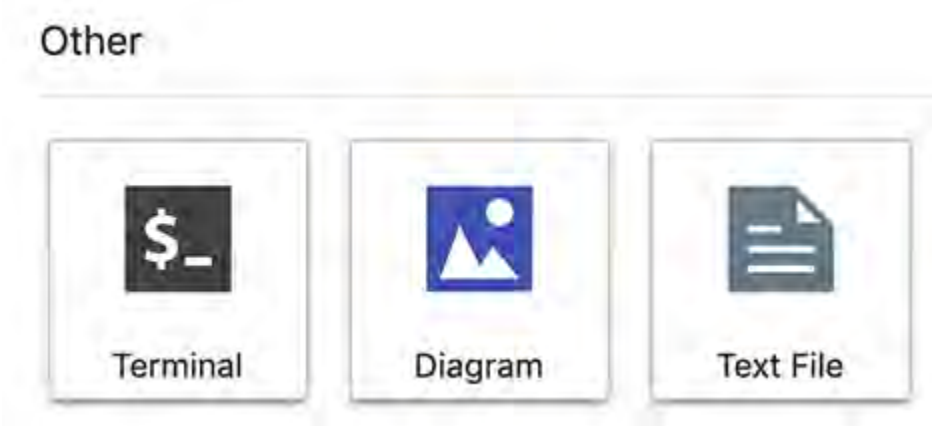


Figura 3.26: Acceso a terminal.

4. Listar archivos en Hadoop

El siguiente comando lista los archivo del directorio especificado:

```
hdfs dfs -ls /user/nobody
```

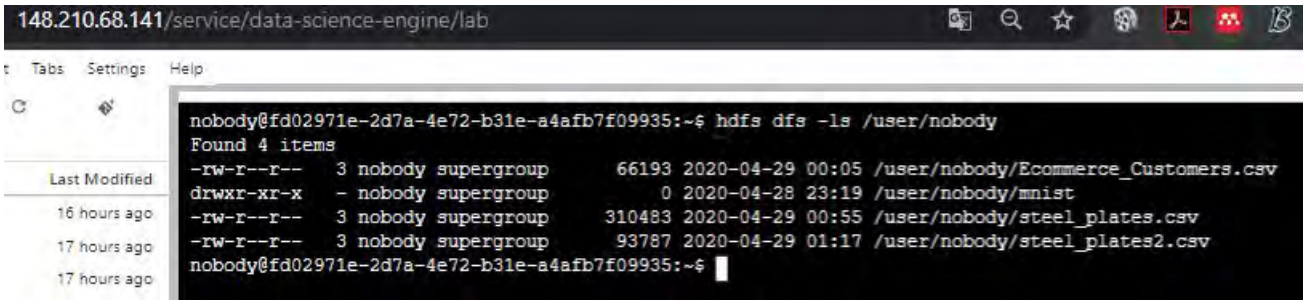


Figura 3.27: Listar archivos Hadoop.

En la Figura 3.27 muestra los archivos listados en el directorio arriba mencionado.

5. Borrar directorio en Hadoop

Para eliminar un directorio el siguiente comando puede ejecutar dicha tarea, incluyendo el borrado de subcarpetas en el directorio.

```
hdfs dfs -rm -R /user/nobody
```

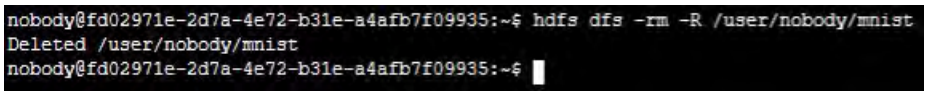


Figura 3.28: Borrar archivos Hadoop.

La Figura 3.28 muestra en la terminal DELETED cuando fue posible borrar un archivo Hadoop.

6. Definir nuevos directorio en Hadoop

Para definir un nuevo directorio el siguiente comando se puede ejecutar en una terminal del DSE.

```
hdfs dfs -mkdir /user/nobody
```

7. Copiar archivos a Hadoop

Este comando se puede describir como el más importante porque permite almacenar datos de gran volumen dentro de Hadoop y se lista como:

```
hdfs dfs -put LOCAL_FILE to  HDFS_PATH
```

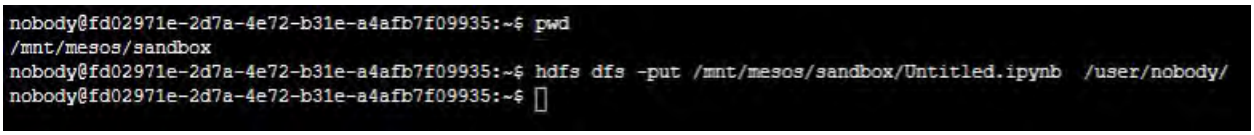


Figura 3.29: Definir nuevo directorio en Hadoop.

En la Figura 3.29 se ejecutó el comando mencionado para mover un archivo llamado Untitled.ipynb a un directorio Hadoop.

8. Validación de Hadoop, Spark y Mlib en DSE

En el siguiente caso de prueba se utilizó un algoritmo de regresión lineal. Disponible en la librería MLib de Spark, predice lo que gastará un cliente en la suscripción de una aplicación de software. El conjunto de datos fue almacenado en Hadoop y el código de programación Python se ejecutó en un libro de notas en la aplicación DSE.

3.6. Prueba de componentes para caso de prueba relacionado al comercio electrónico

Esta actividad fue diseñada para realizar las pruebas iniciales de los *frameworks Big Data*. Para validar el funcionamiento de las librerías de Spark y probar el almacenamiento y funcionalidades de Hadoop, se utilizaron algunos códigos de prueba que se pueden encontrar en repositorios públicos como `Kaggle.com` y `Github.com`.

1. Implementación de la aplicación Customers en Spark

Las siguientes lineas de código permiten conectarse a Spark e iniciar una sesión para definir una aplicación nativa de esta herramienta con el nombre: *Customers*.

```
from pyspark.sql import SparkSession
spark= SparkSession.builder.appName('Customers').getOrCreate()
```

2. Importación del algoritmo de regresión lineal

Las siguientes líneas de código están relacionadas con la importación del algoritmo de regresión lineal de la librería MLib.

```
from pyspark.ml.regression import LinearRegression
```

3. Acceso a Hadoop

El directorio Hadoop se puede acceder de manera automática en el entorno de desarrollo DSE a través de uno de los lenguajes de programación perteneciente al mismo (en este caso, Python). Es decir, en el Capítulo 3, en la etapa de instalación de componentes se configuró el acceso a Hadoop, por lo que, todos los datos almacenados e información se pueden mandar llamar dentro del código siguiente:

```
dataset=spark.read.csv("Ecommerce_Customers.csv",inferSchema=True,header=True)
```

4. Impresión del esquema de datos

El siguiente comando muestra el esquema de tipos de datos que conforman el *dataset* del análisis.

```
 dataset

salida:  DataFrame[Email: string, Address: string, Avg Session

Length: double, Time on App: double,

Time on Website: double,

Length of Membership: double, Yearly Amount Spent: double]
```

5. Mostrar primeros datos del conjunto

En la Figura 3.30 se imprimen los primeros valores del conjunto de datos ejecutando el siguiente comando:

```
dataset.show()
```

Email	Address	Avg Session Length	Time on App
mstephenson@ferna...	835 Frank TunnelW...	34.49726773	12.65565115
hduke@hotmail.com	4547 Archer Commo...	31.92627203	11.10946073
pallen@yahoo.com	24645 Valerie Uni...	33.00091476	11.33027806
riverarebecca@gma...	1414 David Throug...	34.30555663	13.71751367
mstephens@davidso...	14023 Rodriguez P...	33.33067252	12.79518855

Figura 3.30: Primeros valores del conjunto de datos.

6. Árbol de atributos

Para mostrar una estructura básica de árbol con la información de los atributos del *dataset*, como se muestra en la Figura 3.31, ejecutar el siguiente comando:

```
dataset.printSchema()
```

7. Importación de vectores de ensamble

En la regresión lineal es necesario generar un vector de valores dependientes, este se refiere a los atributos de interés involucrados en el análisis, en este caso lo que gastaría un usuario depende de otros atributos como el uso de una aplicación de software, gasto promedio, entre otros, para tal situación se importan las funciones siguientes:

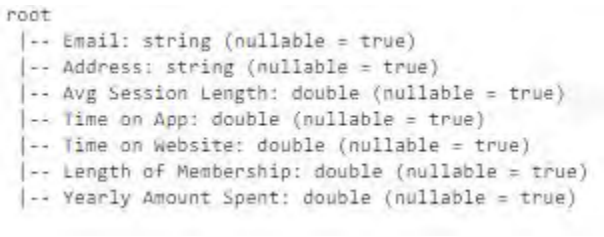


Figura 3.31: Estructura raíz de los datos.

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

8. Asignación de parámetros

Definimos los parámetros del vector de ensamblado, en esta sección es importante la selección de atributos numéricos que pueden estar relacionados con lo que un cliente puede gastar en la aplicación de software. El tiempo que invierte en el uso de esta, el cual puede ser un atributo candidato para esta selección, entre más tiempo pase en una aplicación su interés puede ser mayor en ellas y también su interés en comprarlas. El siguiente comando define dicho vector:

```
featureassembler=VectorAssembler
(inputCols=["Avg Session Length","Time on App","Time on
 Website","Length of"],outputCol="Independent Features")
```

9. Transformación del conjunto de datos

La siguiente linea de código realiza una transformación de Spark a índices, lo que permite mejorar el desempeño del modelo de regresión lineal. Los comandos utilizados son:

```
output=featureassembler.transform(dataset)
```

10. Mostrar datos

A continuación se muestran los datos que fueron transformados con el siguiente comando:

```
output.show()
```

11. Mostrar parámetros independientes

Se pueden mostrar nuevamente los datos seleccionando características independientes.

```
output.select("Independent Features").show()
```

12. Salida de columnas

Este comando muestra columnas de los valores independientes.

```
output.columns
```

13. Selección de salidas de parámetros independientes y parámetros objetivo

```
finalized_data=output.select("Independent Features","Yearly Amount Spent")
```

14. Mostrar datos nuevamente

En este paso se imprimen los datos del paso anterior.

```
finalized_data.show()
```

15. Particionamiento del conjunto de datos para prueba y entrenamiento

```
train_data,test_data=finalized_data.randomSplit([0.75,0.25])
```

16. definición del regresor y entrenamiento del algoritmo

```
regressor=LinearRegression(featuresCol='Independent Features',
labelCol='Yearly Amount Spent')
regressor=regressor.fit(train_data)
```

17. Mostrar coeficientes

El siguiente código imprime los coeficientes de regresión:

```
regressor.coefficients
```

18. Intersección del regresor

El siguiente código intercepta el regresor con el comando:

```
regressor.intercept
```

19. Evaluación de los datos

Se validan los datos para generar la predicción con el comando:

```
pred_results=regressor.evaluate(test_data)
```

20. Resultados de la predicción

En este apartado se muestran las predicciones que indican cuanto gastara un usuario en la suscripción de una aplicación de software contra el monto de lo que compra actualmente, ver Figura 3.32, la predicción se mantiene con valores muy cercanos a lo que gasta, por lo tanto, el modelo de regresión lineal se ajusta a este ejemplo.

```
pred_results.predictions.show(40)
```

En este capítulo se ha descrito el proceso de implementación del clúster y la validación del funcionamiento del mismo a través de un caso de prueba que incluye las librerías de *Machine Learning* de Spark.

```
[41]: pred_results.predictions.show(40)
```

```
+--------------------+-------------------+------------------+
|Independent Features|Yearly Amount Spent|        prediction|
+--------------------+-------------------+------------------+
|[30.57436368,11.3...|        442.0644138| 441.6494190801718|
|[30.73772037,12.6...|        461.7807422| 449.8414199411375|
|[31.30919264,11.9...|        432.7207178| 429.0068859917262|
|[31.3123496,11.68...|         463.591418|444.41918955586357|
|[31.38958548,10.9...|        410.0696111| 409.5710593954009|
|[31.42522688,13.2...|        530.7667187| 533.4836257291336|
|[31.44744649,10.1...|        418.6027421| 427.1385142668919|
|[31.53160448,13.3...|        436.5156057|431.16480053225155|
|[31.57613197,12.5...|         541.226584| 542.5361719871221|
|[31.60983957,12.7...|        444.5455497|426.12582125077984|
|[31.72165236,11.7...|        347.7769266|348.68963238384026|
|[31.81861657,11.2...|        446.4186734| 448.5838435614778|
|[31.86483255,13.4...|        439.8912805| 448.5313264264548|
|[31.8854063,11.28...|         390.103273| 398.9430256056612|
|[31.94539575,12.9...|        657.0199239| 662.3046807365763|
|[32.03054972,12.6...|        594.2744834| 588.5556178168813|
|[32.04448613,13.4...|        448.2298292|445.39667847297824|
|[32.08838063,11.9...|        512.1658664| 518.0297838789022|
|[32.1253869,11.73...|        457.8476959| 437.1493609020465|
|[32.1977238,11.83...|        514.0889577| 508.9649026993302|
|[32.20079864,12.2...|        478.8853913| 472.6222791932512|
|[32.21292383,11.7...|        513.1531119|  513.723712885951|
|[32.25997327,14.1...|        571.2160048| 571.7826498892898|
|[32.27184828,13.4...|          511.97986|505.88985780400867|
|[32.3025531,11.97...|        478.6009159|477.32580076286604|
|[32.31290975,9.82...|        356.6155679| 356.4567484847937|
|[32.33889932,12.0...|        407.7045475| 410.8431656525836|
```

Figura 3.32: Resultados de la predicción.

Capítulo 4

Casos de Prueba

En el presente capítulo se expone el análisis de tres casos de prueba relacionados con la detección de anomalías. El primero, aborda la predicción de retinopatía diabética a través del procesamiento de imágenes y aprendizaje de patrones de la enfermedad con apoyo de una red neuronal convolucional. El segundo análisis, está enfocado a predecir anomalías que generan defectos en la manufactura de placas de acero. El último caso, tiene por objetivo detectar anomalías en las transmisiones mecánicas de turbinas eólicas por medio de sensores de vibración, bajo diferentes condiciones de ruido y vibración. Para todos los casos de prueba, el enfoque de análisis es obtener una comparación entre el desempeño del clúster con los componentes integrados al PaaS y una implementación *standalone* solicitada por un usuario en LANTI. Para las etapas de: adquisición, almacenamiento, preprocesamiento, análisis y construcción de un modelo predictivo, tomando en cuenta cada escenario y condiciones de los datos, tipo, volumen y objetivo del caso.

4.1. Uso general del PaaS para el análisis de casos de prueba

Para acceder al servicio de análisis de datos a través de la plataforma como servicio es necesario ingresar la dirección URL que corresponde con el servicio proporcionado a través del nodo maestro. Este procedimiento se describió en el capítulo 3, actividad 3.3, paso seis. Sin embargo, se puede acceder directamente al entorno de desarrollo Jupyter Labs e interactuar con los *frameworks* de análisis de datos y componentes de *Big Data*. En la Figura 4.1, se ilustra el proceso de comunicación de Jupyter Labs con los componentes instalados del clúster, comunicación que se logra mediante código de programación: *Middleware*. Sin embargo, para monitorear, reiniciar, detener o eliminar servicios es necesario acceder al PaaS como se muestra en la Figura 4.2, este procedimiento, utilizando el navegador de Internet Chrome.

Previamente, un administrador de clúster debe generar las cuentas de usuario con sus respectivos permisos para evitar problemas de acceso. En la Figura 4.3 se ilustra el proceso de ingreso con el usuario que fue registrado con anterioridad.

Una vez realizado el ingreso, localizar el menú de servicios (ver Figura 4.4). Es importante mencionar algunos servicios y componentes instalados en el clúster son inaccesibles por una URL o IP. Es decir, algunos de ellos únicamente permiten acceso vía API. Para ingresar a la plataforma de análisis de datos puede revisar el capítulo 3 en la actividad 3.5 a partir del paso 3. La siguiente tarea necesaria para analizar los datos es definir el proceso de carga y transferencia de la información a un repositorio previamente diseñado.

Uso de *Middleware*

El entorno de desarrollo JupyterLabs tiene la prestación de una terminal integrada (ver Figura 4.5) con el núcleo del sistema operativo Linux. De ese modo, todas las operaciones relacionadas con realizar conexiones a discos en red, copiar datos al clúster Hadoop, descargar conjuntos de datos desde repositorios en en Internet, generar directorios, borrar

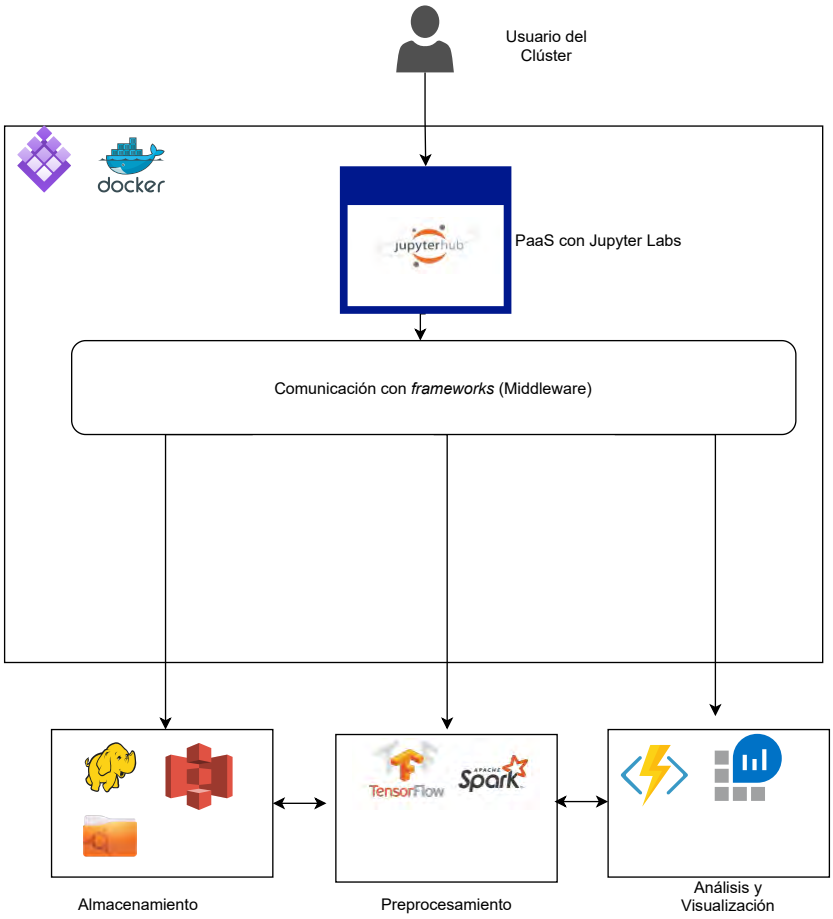


Figura 4.1: Uso del PaaS

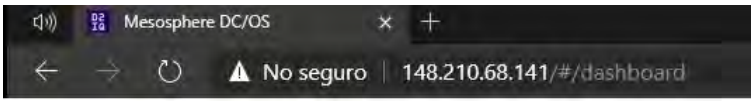


Figura 4.2: Acceso vía dirección IP.

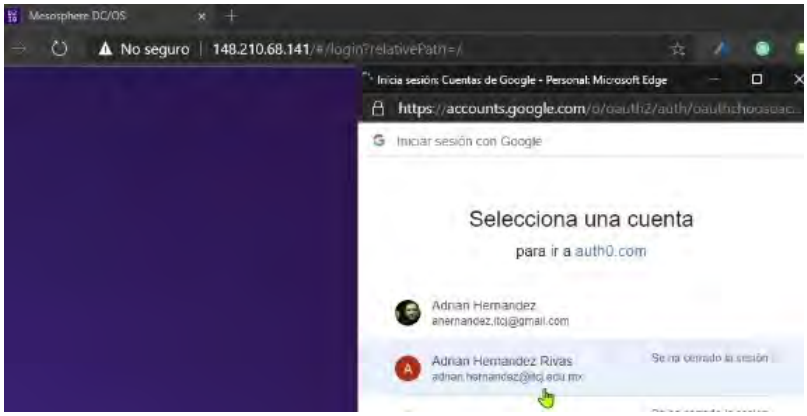


Figura 4.3: Validación de credenciales.

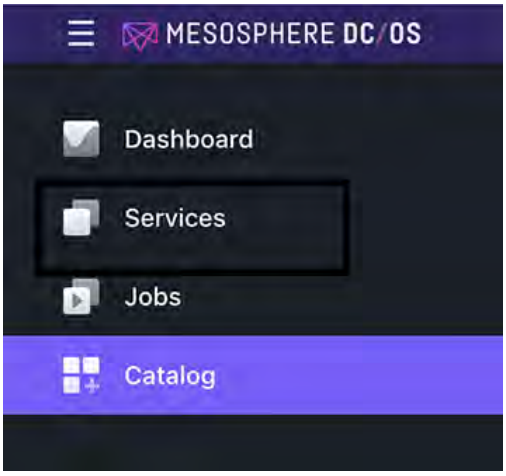


Figura 4.4: Servicios del clúster.

archivos, se pueden ejecutar desde la misma.

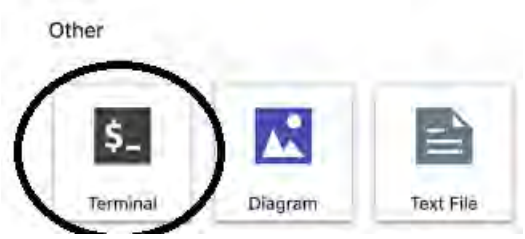


Figura 4.5: Terminal Linux

El siguiente ejemplo descarga directamente de un almacenamiento en nube de un conjunto de datos.

```
wget --no-check-certificate
https://mega.nz/file/H5BjVAIT#4MFLUC-OQnCSZh1JuFdUAbc2tafGO1SqWMIBHwEMWKo
```

En este proyecto se construyó el *Middleware* según el requerimiento de cada una de las pruebas realizadas, por ejemplo: descargar datos de un repositorio de Amazon y copiar esos datos a un clúster Hadoop.

Carga y almacenamiento de la información

Existen varias estrategias para descargar y almacenar la información de diversas fuentes y repositorios externos, según las necesidades del negocio. Entre ellas se mencionan:

Adquisición de datos a clúster Hadoop

La primera, está enfocada en cantidades masivas de datos de repositorios externos de pago, tales como: Amazon, Google, Azure o clústers de base de datos NoSql, por mencionar algunos. En la Figura 4.6, se muestra el diagrama donde se trasladan los datos del repositorio externo a un clúster local Hadoop. Para dicha tarea se utiliza como medio de conexión código *Middleware* escrito en algun lenguaje de programación como Python. En el segmento de código mostrado en la Figura 4.7, la lectura del conjunto de datos ("Customers.csv") se realiza desde el clúster Hadoop. Para consultar más en profundidad sobre el uso Hadoop, en la actividad 3.5, paso 4 se puede profundizar más sobre el uso de comandos. En la Figura 4.7 la función encargada de buscar los datos en Hadoop es:

`Spark.read.csv` (para el caso de archivos csv [1]).

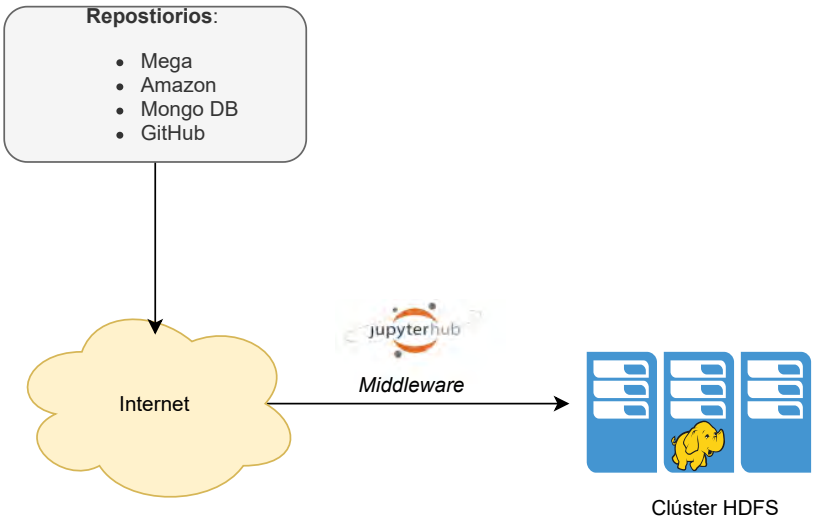


Figura 4.6: Almacenamiento clúster Hadoop.

```
#Ejemplo de uso desde SPARK ACCEDIENDO A HADOOP
#ADRIAN HERNANDEZ RIVAS
from pyspark.sql import SparkSession
spark= SparkSession.builder.appName('Customers').getOrCreate()
from pyspark.ml.regression import LinearRegression
dataset=spark.read.csv("Ecommerce_Customers.csv",inferSchema=True,header=True)
dataset.show()
```

Figura 4.7: Acceso a datos desde clúster Hadoop.

Adquisición a medios locales de almacenamiento

Algunas organizaciones disponen de medios físicos de almacenamiento, tales como discos duros de red, clúster de almacenamiento en red y discos de estado sólido que no tienen el máximo de utilización. En la Figura 4.8 se ilustra el proceso de adquisición de datos desde un repositorio público a medios locales. Es importante mencionar que esta práctica es de utilidad para análisis de datos de poco volumen.

[1] Todo el código mostrado se ejecutó desde el lenguaje Python. Para profundizar en el acceso a Python o cualquier otro lenguaje en el Capitulo 3, 3.5, se describe el proceso completo para ingresar al entorno DSE que es sinónimo de JupyterLabs.

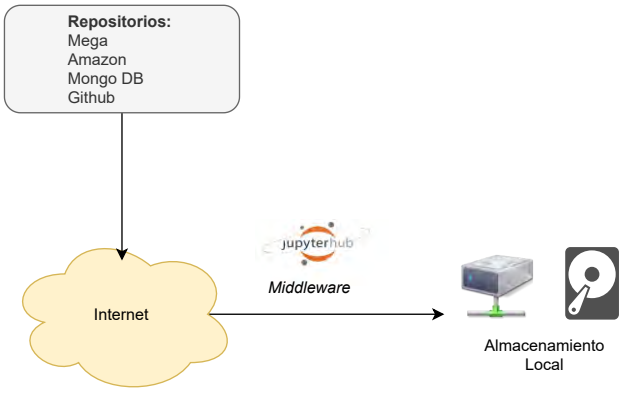


Figura 4.8: Almacenamiento a medios locales.

Lago de datos

Son utilizados para centralizar la información clave del negocio, son conocidos tambien como *Datalakes* (lago de datos, por su significado). En la Figura 4.9 se ilustra el proceso de adquisición de datos desde un lago de datos a medios locales o clúster Hadoop.

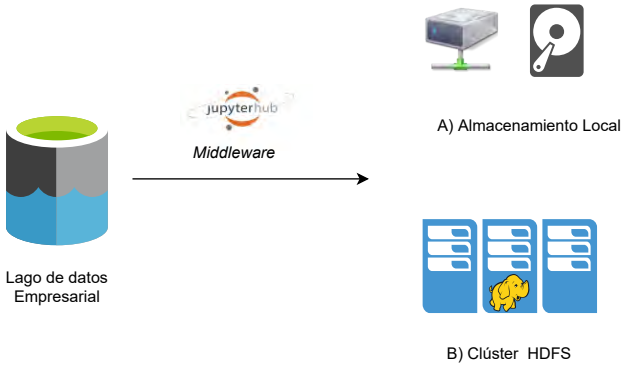


Figura 4.9: Almacenamiento de lago de datos a medios locales o clúster Hadoop.

Origen de repositorios de casos de prueba

Los *dataset* analizados en los casos de prueba presentados en las siguientes secciones son tomados de repositorios públicos accesibles en Internet. En la Tabla 4.1, se publican las url de cada uno de los conjuntos de datos usados en los experimentos llevados a cabo en cada uno de los casos de prueba.

Título del Repositorio	URL
Detección de retinopatía diabética	https://www.kaggle.com/c/diabetic-retinopathy-detection/data
Diagnostico en la caja de engranes en transmisión eólica	https://openei.org/datasets/dataset/gearbox-fault-diagnosis-data
Fallas en placas de acero	https://bigml.com/user/czuriaga/gallery/dataset/50b8c55a035d07198d00007c
Desempeño de líneas de producción Bosch	https://www.kaggle.com/c/bosch-production-line-performance
Datos de consumo personal de aplicaciones de software	https://github.com/krishnaik06/PysparkRegressions/blob/master/Pyspark%20Linear%20Regression.ipynb

Tabla 4.1: Repositorios públicos utilizados en casos de prueba.

4.2. Perspectiva de análisis de los casos de prueba

La visión presentada en los casos de prueba está enfocada a validar el desempeño del clúster y los componentes diseñados para soportar las tareas correspondientes a los procesos de minería de datos. Es decir, presentar varios escenarios bajo los cuales puede ser analizado un mismo caso de prueba pero con diferentes componentes del clúster. Para el análisis se tomaron en consideración la naturaleza de los datos, tipo, cantidad y los posibles componentes para una puesta en producción. Además, otro factor a tomar en cosideración para el análisis es el beneficio al utilizar un PaaS en lugar de una implementación *standalone* solicitada por un usuario del LANTI.

Existe un ciclo de retroalimentación para el proceso del desarrollo, diseño, implementación y pruebas de un proyecto de analítica de datos con enfoque en la detección de anomalías. Este permite la integración de nuevos componentes para mejorar la solución inicial.

En ese sentido, es ideal evaluar si la actividad de agregar nuevas tecnologías puede afectar significativamente el avance de proyecto considerando factores críticos, tales como: presupuesto, tiempo de implementación, recursos adicionales y complejidad técnica. Al igual que el supuesto anterior, se consideran las actividades realizada en LANTI para la implementación de un proyecto. De ese modo, se realiza un análisis para comparar el uso de un Paas desplegado en un clúster y una implementación *standalone* en cada caso de prueba. Entre ellas se mencionan las siguientes:

- solicitud de recursos a través de una requisición escrita a al director y responsables de LANTI.
- Autorización de recursos de infraestructura virtual, cómputo y almacenamiento.
- Desconocimiento de usuario del despliegue de una implementaciones local a una en producción.
- Habilidades y conocimientos técnicos del administrador de la infraestructura en LANTI con respecto a las tecnologías solicitadas por el usuario final.

4.3. Escenario del primer caso de prueba

El primer caso de prueba tiene como objetivo evaluar los componentes candidatos necesarios para implementar un clasificador que permita la predicción de la retinopatía diabética RD. Para tal caso, se evaluarán el escenario de uso del PaaS y una implementación *standalone*. La RD es una afectación derivada de las complicaciones de la diabétes mellitus, afecta los vasos sanguíneos de la retina y da lugar a una baja visión incluso ceguera en el paciente afectado [79].

El diagnóstico clínico de retinopatía diabética RD, es una tarea que involucra la asistencia de uno o varios especialistas médicos experimentados con un alto grado de experiencia. De ese modo, para detectar este padecimiento ocular hay que valorar características muy finas que requieren el apoyo de un sistema de clasificación complejo [80]. En la Figura 4.10 , se ilustran las afecciones de RD según el grado de avance.

En este caso de prueba, se propone un enfoque de redes neuronales convolucionales CNN (por sus siglas en inglés) para diagnosticar RD. De tal modo que, a partir de imágenes a color y con un diagnostico previo, se entrenó una red con arquitectura CNN para un diagnóstico automático y sin intervención del especialista. La muestra se compone por 62 imágenes que se dividen en dos partes: la primera con valoración negativa y la segunda con retinopatía en

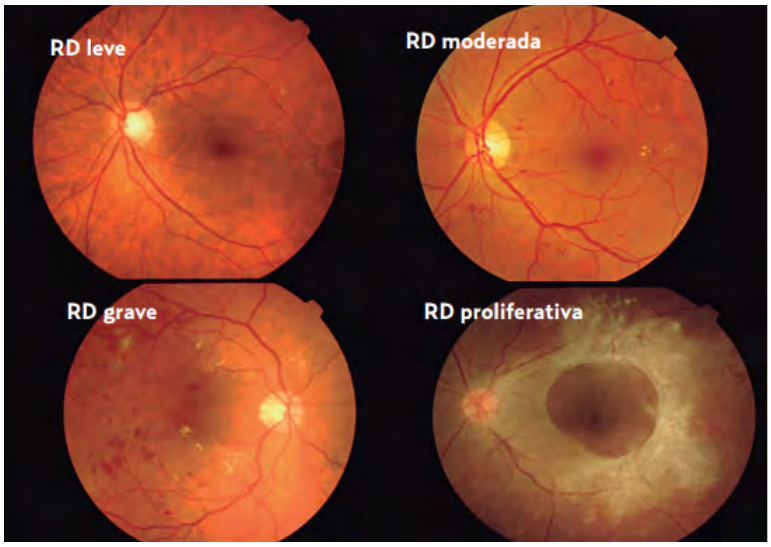


Figura 4.10: Afecciones de retinopatía diabética.

nivel proliferativo.

4.3.1. Conjunto de datos utilizado

Los datos para este caso de prueba están clasificados en imágenes de diversos pacientes en las que se capturó la retina ocular en una alta resolución. Las tomas registraron diferentes condiciones de diagnóstico. Además, para cada paciente se capturó un campo izquierdo y derecho. Las imágenes están etiquetadas con un id de sujeto, Por ejemplo, `1_left.jpeg` es el ojo izquierdo del paciente id 1.

Un médico ha calificado la presencia de retinopatía diabética en cada imagen en una escala de 0 a 4, según la siguiente escala:

- 0 -Sin DR
- 1 -Leve
- 2 - Moderado

- 3 - Grave
- 4 - Proliferante

Según indica la descripción del repositorio [80], las imágenes del conjunto de datos proceden de diferentes modelos y tipos de cámaras, lo que puede afectar a la apariencia visual de izquierda frente a derecha. Además, hay casos en los que las imágenes tienen una presentación tal cual se utiliza en los diagramas anatómicos (mácula a la izquierda, nervio óptico a la derecha para el ojo derecho). Otras se pueden ver como si de por medio interviene una lente condensadora de microscopio (es decir, invertido, como uno ve en un examen típico de ojos vivos). Por lo general, hay dos maneras de saber si una imagen está invertida:

Se invierte si la mácula (la pequeña zona central oscura) es ligeramente más alta que la línea media a través del nervio óptico. Si la mácula es inferior a la línea media del nervio óptico, no se invierte. Si hay una muesca en el lado de la imagen (cuadrado, triángulo o círculo), entonces no está invertida. Si no hay muesca, está invertido.

En conclusión respecto a los datos de origen, el ruido está presente en las imágenes y etiquetas, normalmente como sucede en cualquier conjunto de datos. Adicionalmente, las imágenes pueden contener artefactos, estar fuera de foco, subexpuestas o sobreexpuestas.

4.3.2. Descripción de componentes del clúster utilizados

Para el análisis de este caso de prueba se utilizaron los siguientes componentes y *frameworks*, tales como: Hadoop, permite almacenar los objetos y conjuntos de datos para el análisis ; OpenCv, procesamiento de las imágenes; TensorFlow, para utilizar y activar las funciones de relacionadas con redes neuronales; Numpy, para conversión de imágenes en arreglos numéricos; Python, es un lenguaje de programación que permite desplegar el código de los componentes anteriores. Además, el núcleo de todos los componentes es el entorno de desarrollo Jupyter Labs. En la Figura 4.11 se ilustran los componentes descritos con sus logos

representativos.

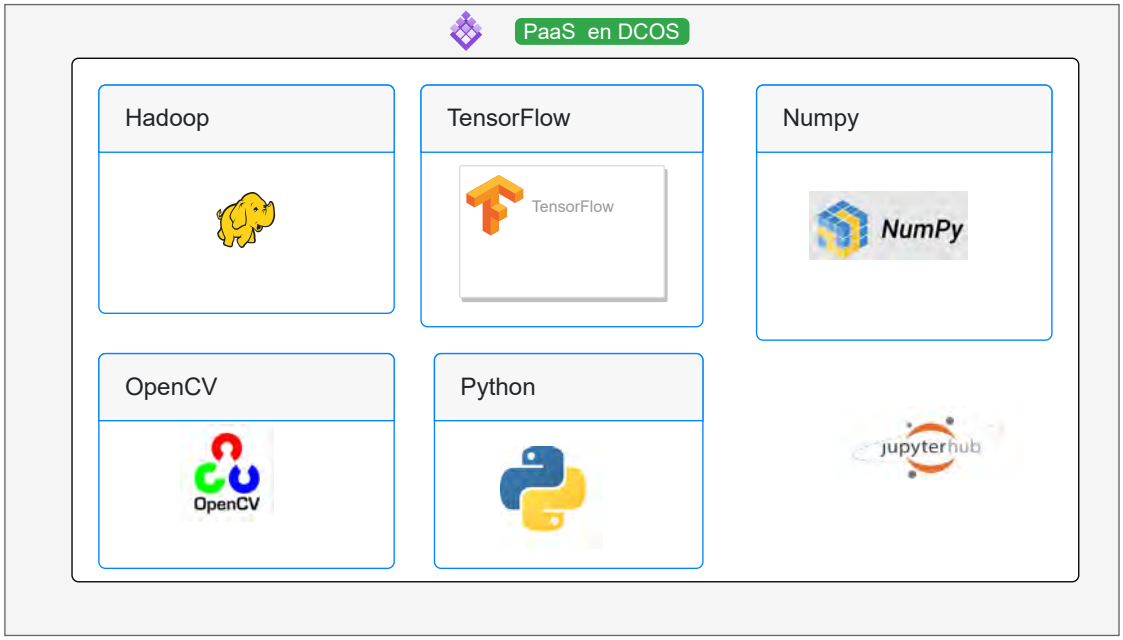


Figura 4.11: Componentes para análisis de RD

4.3.3. Procedimiento básico para la construcción de un clasificador.

El procedimiento para generar un modelo predictivo de análisis de retinopatía diabética en imágenes de alta resolución con el uso del PaaS, se llevó a cabo de la siguiente manera:

- Transferencia de Conjunto de datos a clúster Hadoop. Como primer parte del análisis del caso de prueba, se transfirió el conjunto entero de datos al clúster Hadoop usando el entorno de desrrollo Jupyter, ver Figura 4.12 donde se ilustra el proceso descrito.
- Adquisición de muestra de datos. Hadoop esencialmente tiene por objetivo almacenar grandes volúmenes de datos. Sin embargo, se adquirió una muestra del total del conjunto para probar los componentes de procesamiento de imagen (OpenCV), conversión (Numpy) y análisis con red neuronal (Tensorflow), ver Figura 4.13 donde se ilustra el

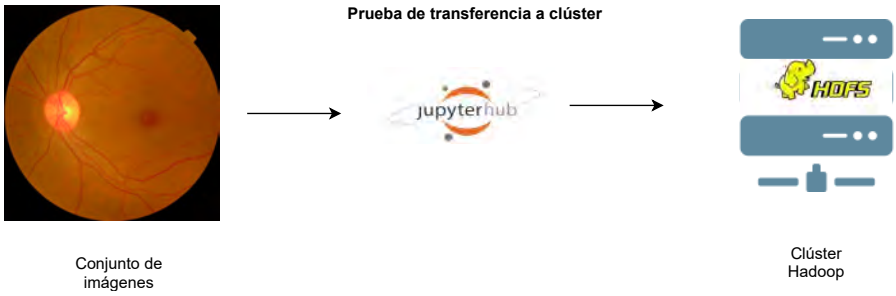


Figura 4.12: Transferencia de imágenes a clúster Hadoop.

proceso descrito.

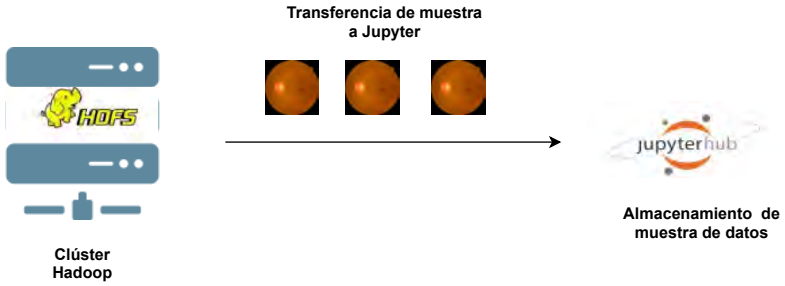


Figura 4.13: Adquisición de muestra de datos.

- Conversión. El proceso de conversión consiste en transformar las imágenes en objetos que puedan ser procesados por una red neuronal convolucional.
- Entrenamiento y análisis con red neuronal convolucional. Esta actividad consistió en la construcción, entrenamiento y validación de un modelo de *Machine Learning*. De ese modo, el primer paso es alimentar la red neruonal para su aprendizaje, este proceso se realiza con imágenes que fueron previamente convertidas en arreglos numéricos hasta un aprendizaje adecuado y finalmente validar el modelo con nuevas elementos que no fueron incluidos durante el entrenamiento del algoritmo de ML

Figura 4.14: Adquisición de muestra de datos.

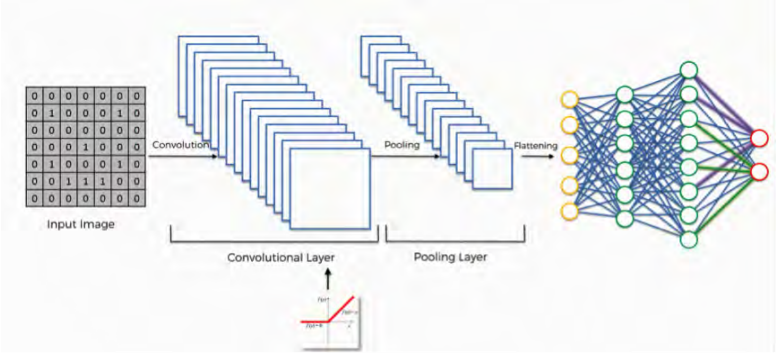


Figura 4.15: Análisis de retinopatía diabética con red convolucional.

4.3.4. Resultados

Los criterios para evaluar la factibilidad del uso del PaaS se describe con los siguientes criterios:

- Factibilidad de Integración. Es necesario evaluar si un componente es factible de incorporar en una arquitectura para analítica de datos en un contexto *Big Data*. Es decir, no es lo mismo utilizar una base de datos relacional dentro de un contexto *Big Data* con el objetivo de mantener una estructura relacional dentro de los datos, comparado a utilizar un componente que permita realizar consultas SQL manteniendo la estructura original *Big Data*.
- Tiempo adicional de implementación. Hace referencia al consumo adicional de horas de trabajo. Es decir, considerando un esquema de contenedores proporcionado por el PaaS propuesto en este reporte de investigación en comparación de un esquema *standalone* donde se parte de únicamente una infraestructura de cómputo y almacenamiento.
- Abstracción de la complejidad. Se refiere a la encapsulación de la complejidad técnica. Este escenario es posible utilizando arquitecturas contenerizadas. Es decir, que no requieren una instalación y configuración paso a paso de librerías, dependencias y servicios del sistema operativo. Sin embargo, en una implementación *standalone* lo anterior tiene que ser llevado a cabo.

Los resultados mostrados en la Tabla 4.2 están relacionados al preprocesamiento, en actividades específicas como transformación y almacenamiento de datos. A continuación se enlistan los resultados mostrados en la gráfica de la Figura 4.16. para las siguientes tareas:

Opciones obtenidas para almacenamiento desde el PaaS:

- Descarga a clúster NoSQL.
- Descarga a volúmen Persistente.
- Descarga a Sistema de Almacenamiento en Red.

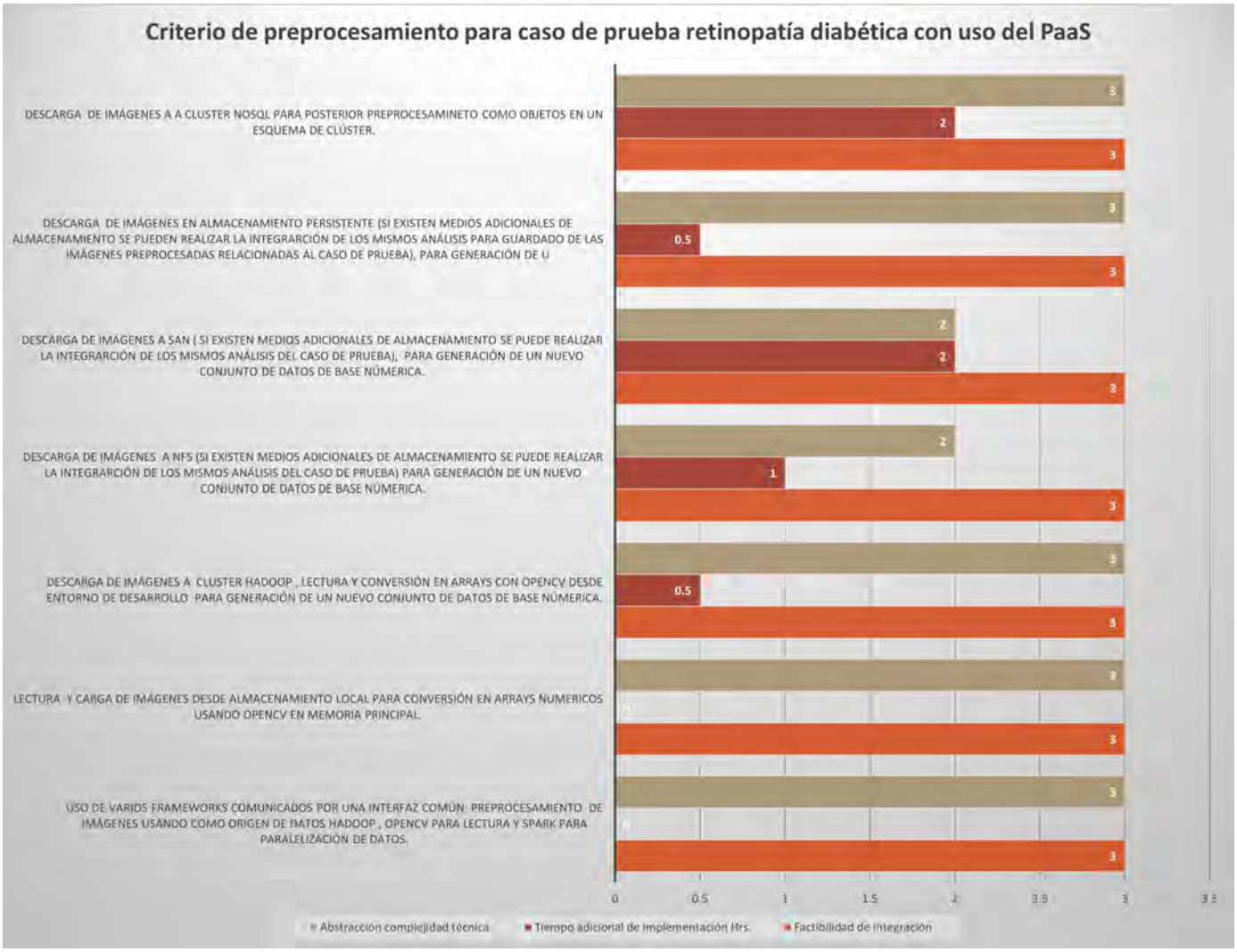


Figura 4.16: Preprocesamiento en caso de retinopatía diabética.

Descripción de la Actividad con integración y uso de componentes del PaaS	Factibilidad integración	Tiempo adicional implementación Hrs.	Abstracción complejidad técnica
Uso de varios frameworks comunicados por una interfaz común: preprocesamiento de imágenes usando como origen de datos Hadoop , OpenCv para lectura y Spark para paralelización de datos.	**3**	**0**	**3**
Lectura y carga de imágenes desde almacenamiento local para conversión en arrays numericos usando OpenCV en memoria principal.	**3**	**0**	**3**
Descarga de imágenes a Cluster Hadoop , lectura y conversión en arrays con OpenCV desde entorno de desarrollo para generación de un nuevo conjunto de datos de base número.	**3**	**0.5**	**3**
Descarga de imágenes a NFS (si existen medios adicionales de almacenamiento se puede realizar la integrarción de los mismos análisis del caso de prueba) para generación de un nuevo conjunto de datos de base número.	**3**	**1**	**2**
Descarga de imágenes a SAN (si existen medios adicionales de almacenamiento se puede realizar la integrarción de los mismos análisis del caso de prueba), para generación de un nuevo conjunto de datos de base número.	**3**	**2**	**2**
Descarga de imágenes en almacenamiento persistente (si existen medios adicionales de almacenamiento se pueden realizar la integrarción de los mismos análisis para guardado de las imágenes preprocesadas relacionadas al caso de prueba), para generación de un nuevo conjunto de datos de base número	**3**	**0.5**	**3**
Descarga de imágenes a a cluster NoSQL para posterior preprocesamineto como objetos en un esquema de clúster.	3	2	3

Tabla 4.2: Tareas de preprocesamiento, transformación y almacenamiento de datos.

- Descarga a Sistema de Archivos en Red.
- Descarga de imágenes a clúster Hadoop.

Opciones para el preprocesamiento y tratamiento de imágenes:

- Lectura y carga de imagenes desde almacenamiento local.
- Combinación de *frameworks* en una interfaz común.

Resultados con uso del PaaS

Los resultados mostrados en este apartado, muestran mejoras en los criterios de tiempo adicional de implementación y abstracción de complejidad para despliegue de nuevos componentes en comparación a la implementaciómn *standalone*.

Resultados sin uso del PaaS

El resultado general para tiempos adicionales de implementación es mayor en comparación al uso del PaaS, ver Figura 4.17. Sin embargo, otro factor que se debe tomar en consideración es la abstracción de la complejidad, en este apartado también es mayor.

Descripción de la Actividad con integración y uso de componentes del PaaS	Factibilidad integración	Tiempo adicional implementación Hrs.	Abstracción complejidad técnica
Uso de varios frameworks comunicados por una interfaz común: preprocesamiento de imágenes usando como origen de datos Hadoop , OpenCv para lectura y Spark para paralelización de datos.	**3**	**504**	**0**
Lectura y carga de imágenes desde almacenamiento local para conversión en arrays numericos usando OpenCV en memoria principal.	**3**	**0**	**3**
Descarga de imágenes a Cluster Hadoop , lectura y conversión en arrays con OpenCV desde entorno de desarrollo para generación de un nuevo conjunto de datos de base númerica.	**3**	**504**	**0**
Descarga de imágenes a NFS (si existen medios adicionales de almacenamiento se puede realizar la integrarción de los mismos análisis del caso de prueba) para generación de un nuevo conjunto de datos de base númerica.	**3**	**2**	**2**
Descarga de imágenes a SAN (si existen medios adicionales de almacenamiento se puede realizar la integrarción de los mismos análisis del caso de prueba), para generación de un nuevo conjunto de datos de base númerica.	**3**	**2**	**2**
Descarga de imágenes en almacenamiento persistente (si existen medios adicionales de almacenamiento se pueden realizar la integrarción de los mismos análisis para guardado de las imágenes preprocesadas relacionadas al caso de prueba), para generación de un nuevo conjunto de datos de base númerica	**3**	**504**	**0**
Descarga de imágenes a a cluster NoSQL para posterior preprocesamineto como objetos en un esquema de clúster.	3	504	0

Tabla 4.3: Tareas de preprocesamiento, transformación y almacenamiento de datos sin PaaS.

Figura 4.17: Preprocesamiento en caso de retinopatía diabética sin uso del PaaS.

4.4. Escenario del segundo caso de prueba

Para este caso de prueba se utilizó un conjunto de datos para predecir fallas de calidad en la manufactura de placas de acero. La clasificación considera siete defectos de calidad en las placas de acero. El objetivo es utilizar un algoritmo de *Machine Learning* para el reconocimiento automático de patrones y saber qué tipo de falla tendrá el producto cuando se presenta una variación anómala en los parámetros del proceso de manufactura.

4.4.1. Conjunto de datos utilizado

El conjunto de datos analizado consta de 1,941 registros, y 28 campos, de los cuales tres son categóricos y 25 numéricos. Los tipos de falla en las placas de acero identificados son: golpes, suciedad, calidad, rasguño-Z, rasguño-K, manchas, otras fallas. En la Figura 4.18, tomada del repositorio UCI[2] , muestra características complementarias de los datos, tales como: número de atributos, valores faltantes.

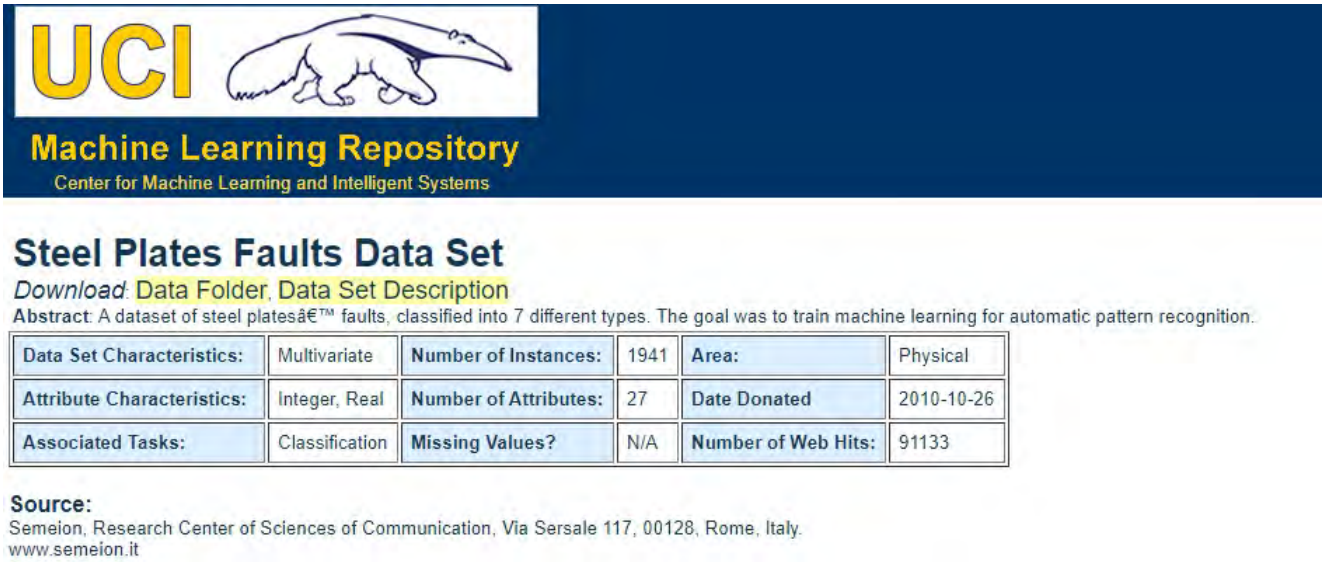
4.4.2. Descripción de los componentes utilizados

Para este caso de prueba se realizó un almacenamiento en Hadoop y un procesamiento con Spark y en el análisis utilizando librería SparkML. En un segundo análisis se utilizó python como lenguaje de programación; SciKit, librería cientifica con los algoritmos ML.

4.4.3. Procedimiento

El procedimiento para el análisis de placas de acero se realizó bajo el siguiente esquema de actividades para una prueba local de librerias, entre ellas:

[2] Figura tomada de: `https://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults`



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Steel Plates Faults Data Set

Download: Data Folder, Data Set Description

Abstract: A dataset of steel platesâ€™ faults, classified into 7 different types. The goal was to train machine learning for automatic pattern recognition.

Data Set Characteristics:	Multivariate	**Number of Instances:**	1941	**Area:**	Physical
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	27	**Date Donated**	2010-10-26
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	91133

Source:
Semeion, Research Center of Sciences of Communication, Via Sersale 117, 00128, Rome, Italy.
www.semeion.it

Figura 4.18: Descripción del conjunto de datos de placas de acero.

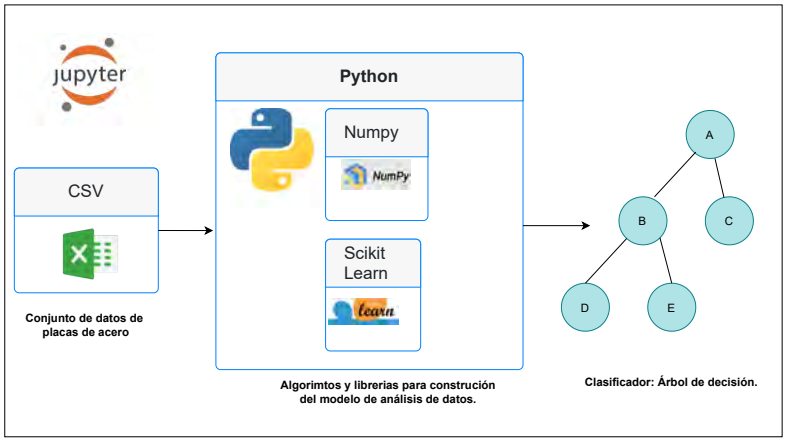


Figura 4.19: Descripción ilustrativa del análisis de datos.

- Carga de archivo CSV a Júpiter
- Uso de Python como lenguaje de desarrollo.
- Definición y uso de librerías de apoyo para la construcción del modelo clasificador, tales como: Scikit Learn, Numpy, Pandas.
- Entrenamiento, validación y pruebas del clasificador.
- Conexión con clúster de prueba y producción.

Para efectos de análisis de este caso de prueba el escenario propuesto es un ambiente de producción en la industria de manufactura de placas de acero. Generalmente, hay un sistema de generación de datos en crudo sobre los procesos de producción [31]. Posteriormente se requiere una adquisición, preprocesamiento y almacenamiento como primer procedimiento. En dicha situación, lo recomendable es la construcción de un pipeline, es decir código de soporte para automatizar los procesos mencionados.

4.4.4. Resultados

En las Tablas 4.4 y 4.5, se presentan los criterios de comparación que evalúan la factibilidad de integración, tiempo adicional de implementación y abstracción de la complejidad en relación a las actividades necesarias para resolver el caso de prueba presentado en un esquema de producción y manejo de *Big Data*.

Con uso del PaaS

El resultado de una implementación mediante uso del PaaS permite las siguientes ventajas: *frameworks* comunicados por una interfaz común, cómputo distribuido, gestión centralizada con la opción de implementar un clúster en producción y uno de prueba en el mismo entorno de PaaS. Además, este último cuenta con una capa de aislamiento de recursos. Es decir, el funcionamiento de los servicios y recursos no interfieren entre sí. De ese modo, a continuación se presentan las soluciones de los procesos de analítica de

Descripción de la Actividad con integración y uso de componentes del PaaS	Factibilidad integración	Tiempo adicional implementación Hrs.	Abstracción complejidad técnica
Desarrollo e integración de Pipelines ML basados en especificaciones de aplicación a gran escala	3	0	3
Desarrollo e Integracion de Pipelines ML paralelos	3	0	3
Migración a proveedoores de nube pública	3	0	3
Escalabilidad horizontal y vertical para puesta en producción	3	0	3
Conexión con clúster de prueba y producción.	3	0	3

Tabla 4.4: Análisis de placas de acero con uso del PaaS

Descripción de la Actividad con integración y uso de componentes sin uso del PaaS	Factibilidad integración	Tiempo adicional implementación Hrs.	Abstracción complejidad técnica
Desarrollo e integración de Pipelines ML basados en especificaciones de aplicación a gran escala	3	15	1
Desarrollo e Integracion de Pipelines ML paralelos	3	60	1
Migración a proveedoores de nube pública	3	30	1
Escalabilidad horizontal y vertical para puesta en producción	3	5	1
Conexión con clúster de prueba y producción.	3	60	1

Tabla 4.5: Análisis de placas de acero a gran escala sin uso del PaaS

datos para solucionar el caso de prueba presentado. Entre ellas se mencionan:

- Construcción de pipelines ML basados en especificaciones de aplicación a gran escala.
- Desarrollo e integración de pipelines ML paralelos.
- Migración a proveedores de nube pública
- Escalabilidad horizontal y vertical para puesta en producción.
- Conexión con clúster de prueba y producción.

Sin uso del PaaS

A diferencia de la implementación con uso del PaaS técnicamente el cómputo distribuido sobre las implementaciones no es posible en el esquema de *standalone*. Sin embargo, opciones como: un clúster de prueba y producción, es factible pero es aqui donde inicia el proceso de complejidad por factores como: tiempo de implementación, configuración y comunicación de *frameworks*.

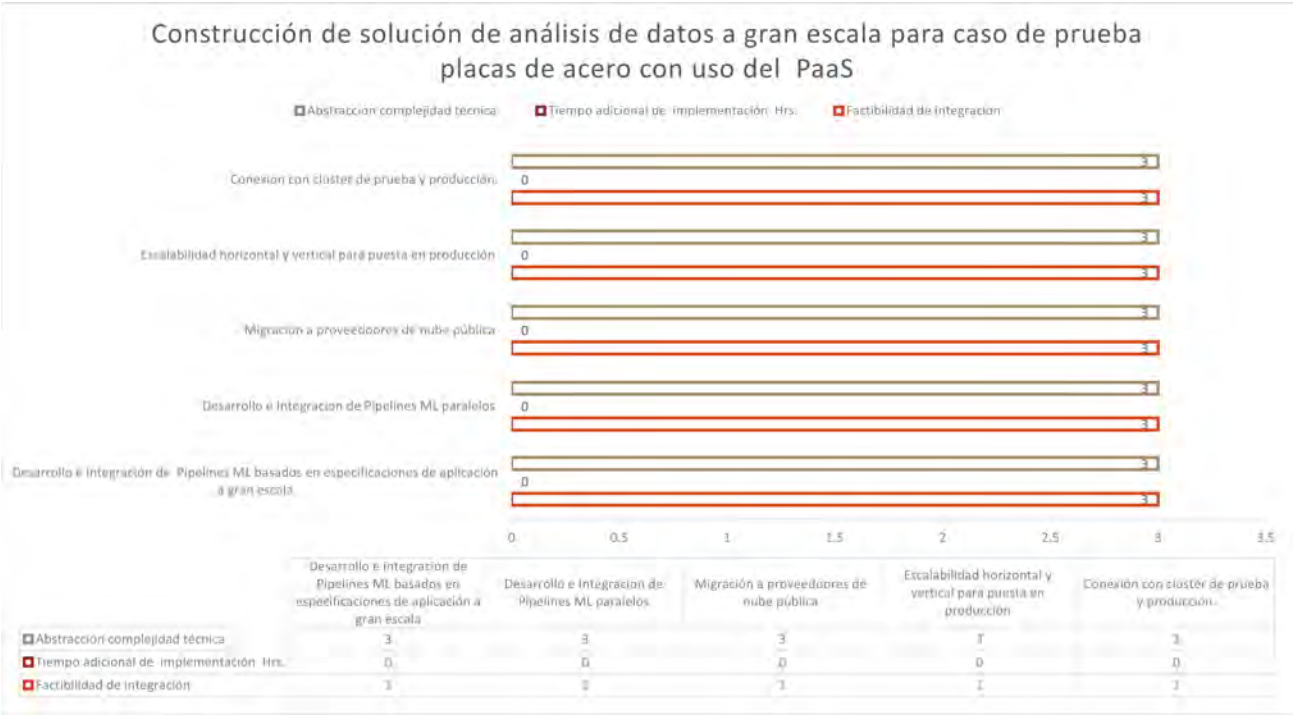


Figura 4.20: Análisis a gran escala caso de prueba de placas de acero en manufactura.

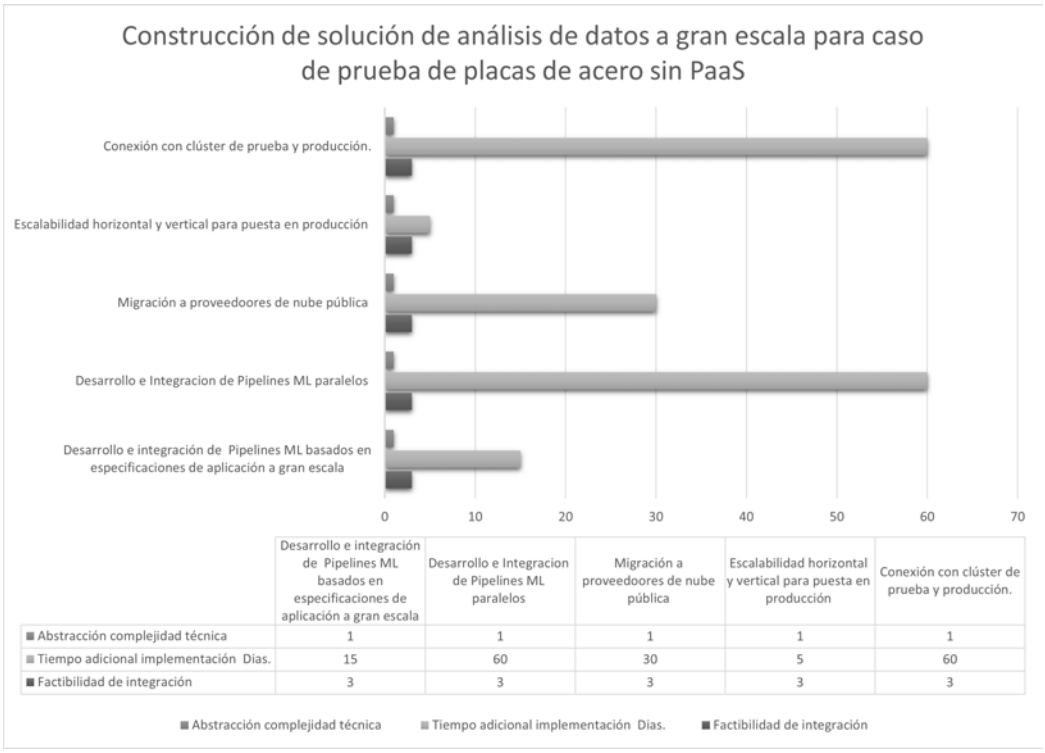


Figura 4.21: Análisis a gran escala de placas de acero en manufactura sin uso del PaaS.

4.5. Escenario del tercer caso de prueba

Las turbinas eólicas modernas [3] , son utilizadas para la generación de electricidad principalmente. Están constituidas internamente por un subsistema de transmisión mecánica, de tal manera qué el movimiento transmitido desde el viento a la turbina es convertido en energía eléctrica.

Sin embargo, cuando se presentan anomalías en el funcionamiento de las turbinas se debe a engranes rotos dentro del subsistema de transmisión mecánica. Para predecir y detectar la falla mencionada y evitar el desarrollo de anomalías de funcionamiento en el generador eólico, se analizó una muestra de un conjunto de datos con el registró de las vibraciones en diferentes condiciones de carga (velocidades). De ese modo, para este caso particular fue seleccionada una carga del 90 %, con cuatro sensores colocados en distintas posiciones dentro de la caja (norte-sur, este-oeste) y a 30 Hz de frecuencia de grabación. Además, las dos variables que representan el estado de la caja son: 0) Condición saludable y 1) Condición de diente roto.

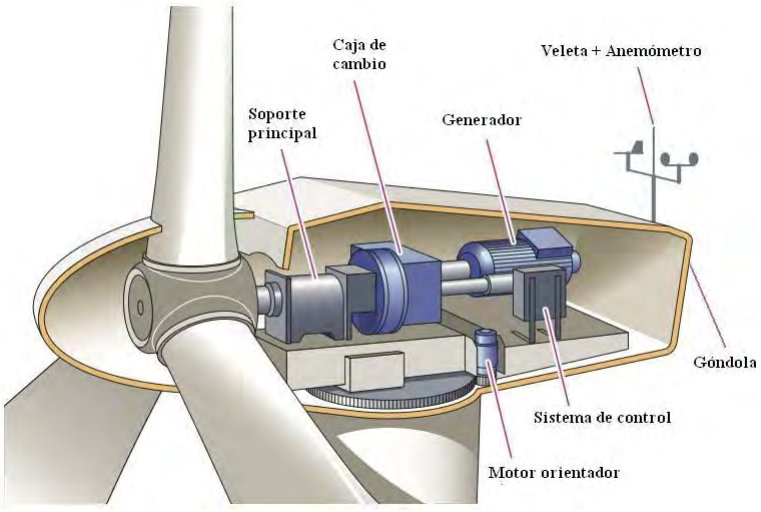


Figura 4.22: Turbina eólica.

[3] imagen tomada de: `https://bit.ly/3yLWnn7`

Escenario de análisis

Almacenamiento, es el escenario considerado para analizar los componentes del cluster que pueden contribuir a la implementación de este caso de prueba. Es decir, bajo el supuesto de una implementación en producción. Además, en la descripción del conjunto de datos, el tipo de archivos generados son de texto y generados por separado segun corresponda con la variable de carga.

4.5.1. Conjunto de datos utilizado

Conformado por diez archivos de texto. El total de instancias en condiciones sin fallas son: 1,015,808 (10 archivos txt), en el caso con falla (dientes de engrane rotos) son un total de instancias: 1,005,311 (10 archivos txt).

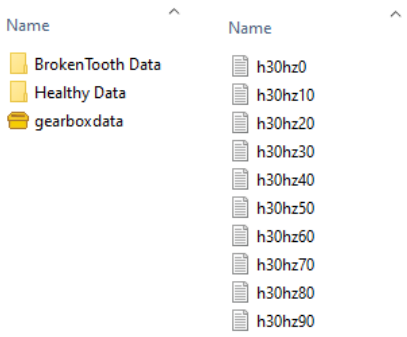


Figura 4.23: Conjunto de datos relacionados con las turbinas eólicas.

En la Figura 4.23 se ilustran los datos originales del conjunto completo. Es importante mencionar que los datos fueron procesados a un nuevo datagrama en formato separado por comas previo al análisis llevado a cabo.

4.5.2. Descripción de componentes utilizados

Entre los componentes utilizados para el análisis de este caso de prueba a una escala de muestra se incluyen los siguientes:

- Almacenamiento local. Para el preprocesamiento de los datos se utilizó almacenamiento local. Es decir, no fue requerido el uso de un clúster de almacenamiento como Hadoop por las dimensiones de memoria del conjunto de datos original y modificado.
- Datagramas usando pandas
- Red Neuronal Perceptron con el componente Keras

Componentes para un escenario en producción relacionado con almcaneamiento

Para un escenario en producción se recomienda:

- Almacenamiento en clúster Hadoop. Permite almacenar millones de archivos.
- Procesamiento paralelo con spark. Con spark se pueden generar conjuntos de datos de forma organizada y guardarse en un esquema con mayor jerarquización en comparación a los archivos de texto por separado.
- Generación de datagramas resilentes. Este esquema forma parte del preprocesamiento y es eficiente con el componente spark por la forma en que los datos son procesados paralelamente.

4.5.3. Procedimiento para construcción del clasificador

Para definir la red neuronal se utilizó Keras, dicha librería permite utilizar funciones de redes neuronales a través de Tensorflow. Entre las múltiples opciones, de *frameworks* de redes neuronales, seleccionamos este último por el robusto soporte que ha tenido durante los últimos años. Primero, se define el modelo secuencial y se agregan las capas de una en una hasta obtener una arquitectura inicial de red, considerada adecuada. Lo primer capa de

entrada, debe definirse con el número correcto de variables de entrada, esto es variable, según cada estudio de caso y sus respectivas entradas y salidas.

La primera capa de entrada se define con el parámetro: `input_dim` y se establece el número de cuatro entradas, esto corresponde con las variables de lectura de cada sensor. Las capas completamente conectadas se definen usando la clase Densa. En dicha clase se pueden especificar el número de neuronas o nodos en cada capa. Como primer argumento, recibe la función de activación de unidad lineal rectificada denominada ReLU en las dos primeras capas y la función Sigmoide en la capa de salida.

La función Sigmoide se utiliza en la capa de salida para garantizar que los parámetros estén en el rango de valores 0 y 1. La arquitectura de la red, consta de una capa oculta con 12 nodos, la segunda capa oculta tiene 8 nodos, además función de activación Relu. En la Figura4.24, se ilustra la arquitectura descrita de la Red Neuronal Artificial perceptron de una tres capas.

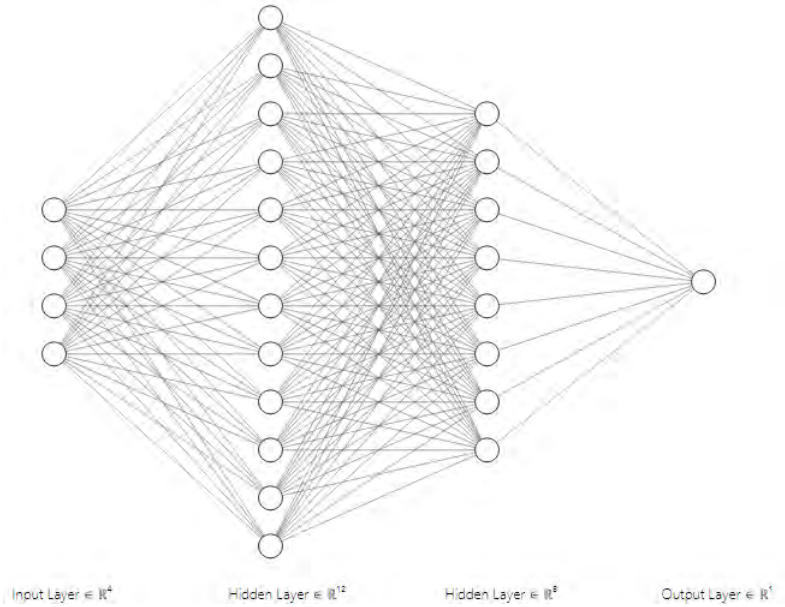


Figura 4.24: Arquitectura de la Red Neuronal utilizada para caso de turnina eólica.

4.5.4. Resultados

Los resultados mostrados en la Tabla 4.6 y Tabla 4.7, son relacionados a los componentes para almacenamiento de datos, este proceso se compara desde dos perspectivas, la primera tomando en consideración el uso del PaaS y la segunda una implementación *standalone* solicitada por un usuario en Lanti. La representación gráfica de estos mismos resultados se muestra en las Figura 4.25 y 4.26.

Sin embargo, a manera de resumen se enlistan las características evaluadas de almacenamiento que corresponden a las tablas y figuras mencionadas, tales como:

- ClusterNoSQL
- Almacenamiento persistente
- Descarga a SAN
- Descarga a NFS
- Descarga a clúster Hadoop
- Descarga a directorio local

Resultados con PaaS

- El clúster es compatible con medios de almacenamiento modernos que usan el protocolo NVMe. Este último ofrece un mejor rendimiento y con cargas de trabajo más breves.
- La integración y comunicación de componentes bajo una interfaz común permite comunicar frameworks que contribuyen a tareas como la generación y compactación de los datos que en en su fuente original son de texto a conjuntos de datos resilentes distribuidos.
- Conexión inmediata con clúster Hadoop.
- El PaaS utiliza un esquema compatible con contenedores por lo que utilizar un almacenamiento de medios persistentes resulta favorable.

Actividad	Factibilidad de integración	Tiempo adicional implementación Hrs.	Abstracción complejidad técnica
Descarga a Directorio Local	**3**	**0.16**	**3**
Descarga a Cluster Hadoop	**3**	**0.5**	**3**
Descarga a NFS	**3**	**1**	**2**
Descarga a SAN	**3**	**2**	**2**
Almacenamiento persistente	**3**	**0.33333**	**3**
Cluster NoSQL	3	0.5	3

Tabla 4.6: Caso de prueba turbinas eólicas con uso del PaaS

- Sí y solo cuando es necesario migrar el contenido de almacenamiento se puede realizar inmediatamente bajo un esquema de contenedores para medios persistentes.

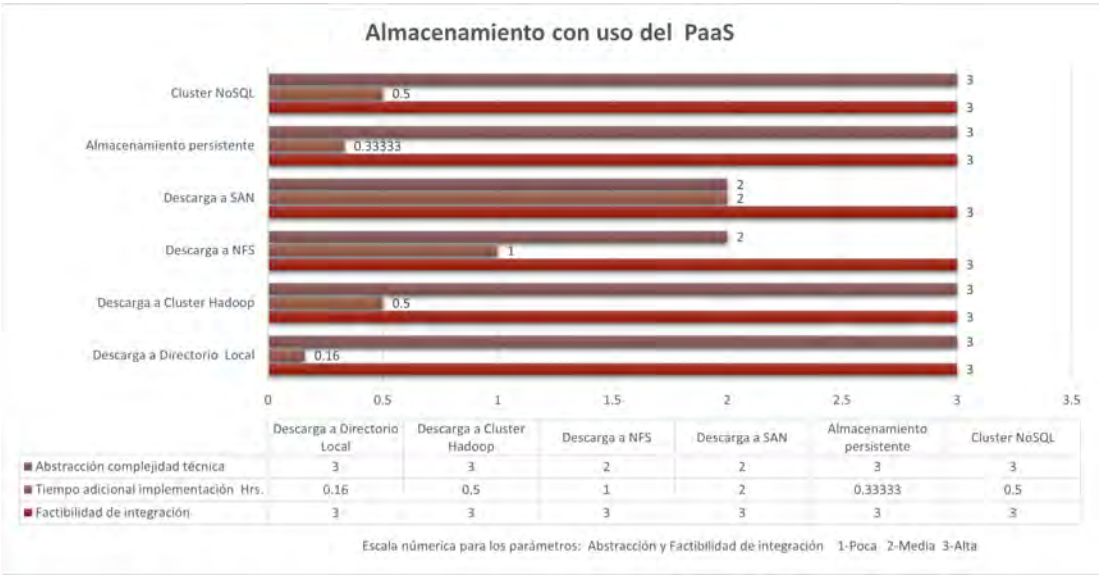


Figura 4.25: Análisis de turbinas eólicas con uso del PaaS

Resultados sin PaaS

- Esquema de almacenamiento descentralizado.
- Compatibilidad con la mayoría de los criterios mostrados en la Tabla 4.7. Sin embargo, los dos factores críticos en este proceso son la abstracción de complejidad y tiempos adicionales de implementación.
- Infraestructura migrable a través del enfoque de máquinas virtuales.

Actividad	Factibilidad de integración	Tiempo adicional implementación Hrs.	Abstracción complejidad técnica
Descarga a Directorio Local	**3**	**0.16**	**3**
Descarga a Cluster Hadoop	**3**	**504**	**1**
Descarga a NFS	**3**	**1**	**3**
Descarga a SAN	**3**	**2**	**3**
Almacenamiento persistente	**3**	**504**	**1**
Cluster NoSql	3	504	1

Tabla 4.7: Caso de prueba turbinas eólicas sin uso del PaaS

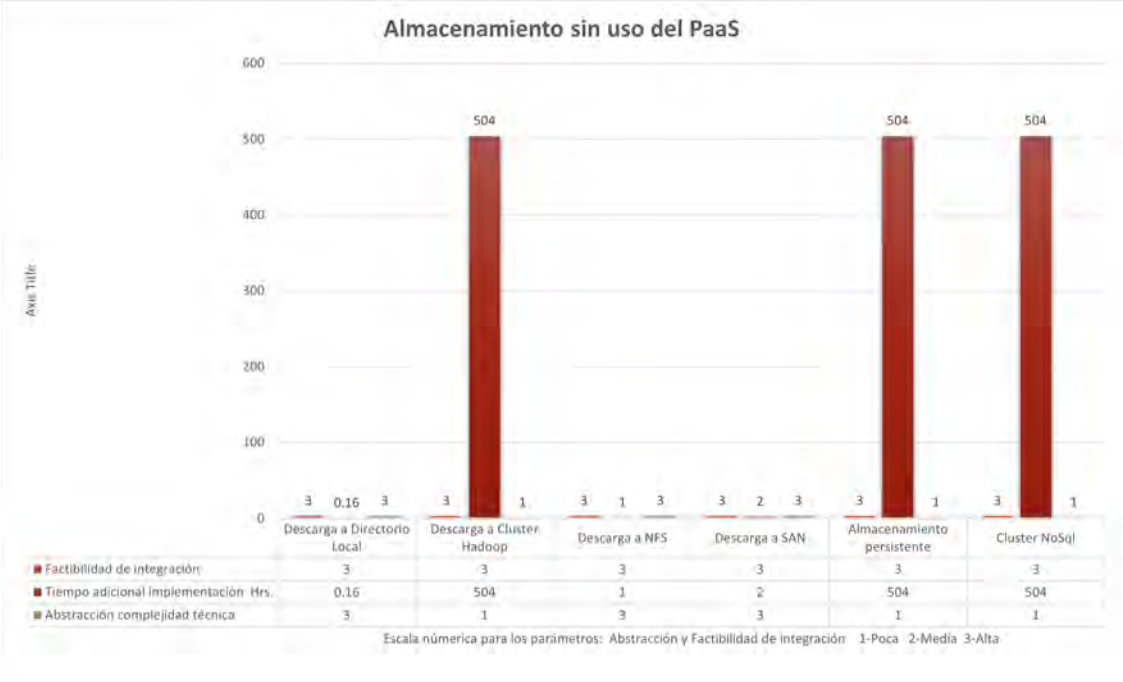


Figura 4.26: Análisis de turbinas eólicas sin uso del PaaS

Capítulo 5

Resultados y Discusiones

Como resultado de este trabajo se cuenta con un clúster de cómputo de alto rendimiento, que es la base de la plataforma para el desarrollo de proyectos *Big Data*. Es compatible con arquitecturas y componentes en un esquema de contenedores usando la tecnología Docker. La administración se realiza a través del modelo de servicio de PaaS. En la Figura 5.1, se ilustra la interfaz web de administración de los distintos servicios del PaaS como el tablero principal (Dashboard), servicios en ejecución (Services), redes (Networking) y clúster (Cluster) por mencionar algunos.

Se cuenta además, a la conclusión de este proyecto, con una plataforma diseñada y configurada para realizar análisis de datos con enfoque en la detección de anomalías. Dicha plataforma se construyó con la intención de ponerla a disposición de usuarios del Laboratorio Nacional de Tecnologías de la Información, la mayoría de ellos investigadores, en la modalidad de plataforma como servicio.

Esta plataforma cuenta con los componentes básicos para ejecutar tareas relacionadas con *Big Data* y analíticas avanzadas de datos. A fin de comprobar el correcto funcionamiento y la utilidad de la plataforma, estos componentes se utilizaron de forma conjunta para lograr realizar el análisis de tres casos de prueba relacionados con la detección de anomalías. Cada uno de ellos con ciertas particularidades, que permitieron evaluar el uso de diversas herramientas

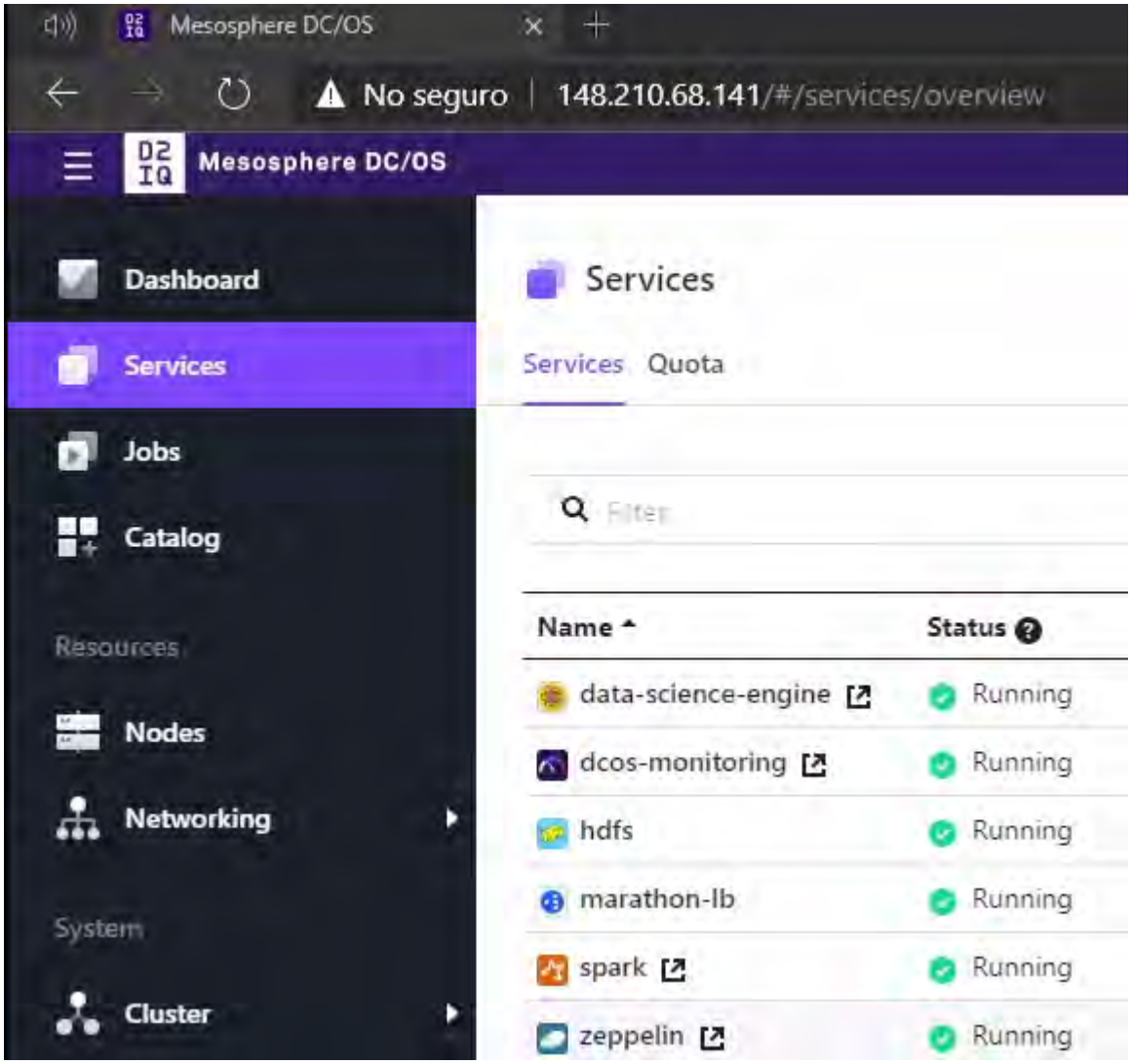


Figura 5.1: Clúster para ejecución de sistemas *Big Data*.

y *frameworks*. Por ejemplo, para visión por computadora, OpenCV; almacenamiento externo y en clúster Amazon S3 y Hadoop; redes neuronales Tensorflow, por mencionar algunos. En la Figura 5.2, se muestran algunos iconos de las herramientas integradas en el entorno de desarrollo Jupyter Labs, utilizado para validar los componentes mencionados.

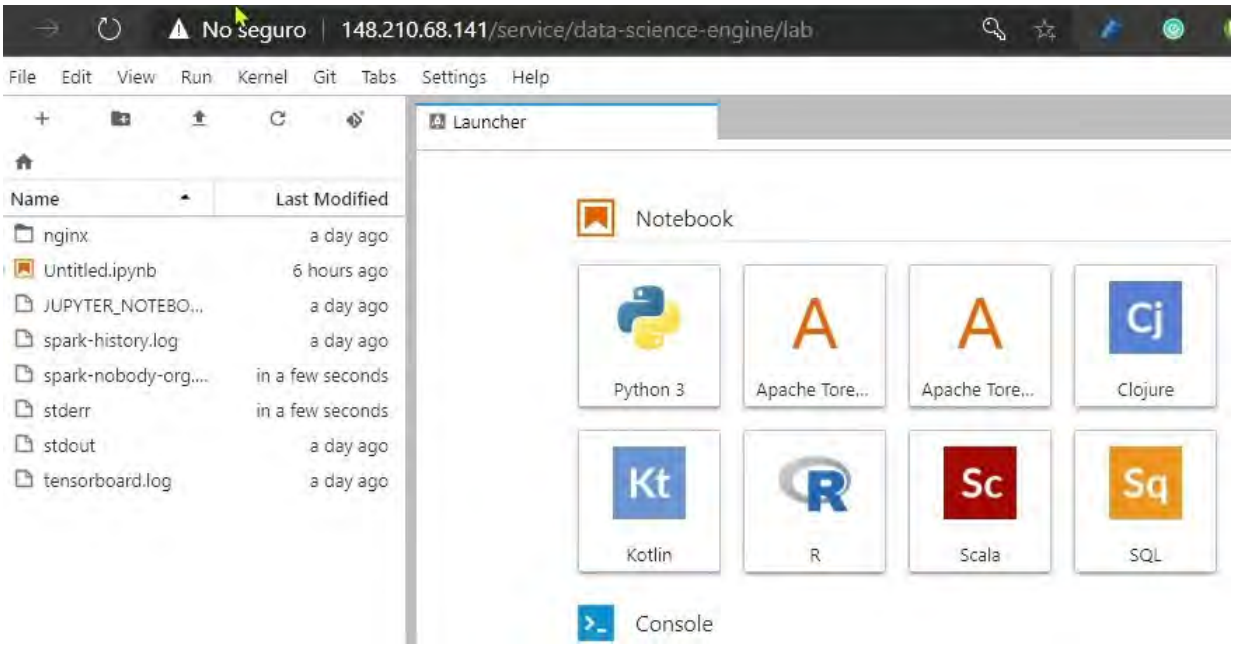


Figura 5.2: Instalación e integración de componentes y *frameworks* para el proceso *Big Data*.

Para el primer caso relacionado con la predicción de retinopatía diabética, se verificó que aunque el componente de redes neuronales funcionó correctamente, hay factores que no se relacionan con el clúster, tal como: ruido en la imagen, ángulo y resolución en la que fue tomada, por mencionar algunos. De ese modo, se obtuvieron malas predicciones por circunstancias externas al rendimiento y desempeño propio del clúster. En la Figura 5.3, se muestra una imagen con los resultados positivos (`YO`) y negativos (`NO`) acerca del padecimiento de retinopatía diabética severa. Este clasificador se construyó con una red neuronal convolucional.

Para el segundo caso de prueba relacionado a la detección de anomalías en placas de acero bajo un esquema de producción en manufactura se confirmó que los tiempos de implemen-

Figura 5.3: Diagnóstico de retinopatía con Red Neuronal Convolucional.

tación y complejidad de la configuración se reducen considerablemente usando el modelo de servicio PaaS. Esto último fue verificado a través de las pruebas realizadas con dos diferentes *frameworks* de aprendizaje automático: SparkML y Scikit-Learn. Ambos fueron utiizados en el mismo entorno de desarrollo, comprobando que el investigador puede centrarse más en la lógica de la aplicación utilizando un entorno centralizado de PaaS.

En el tercer caso de prueba relacionado con el análisis de anomalías en turbinas eólicas, se confirma que el procesamiento y predicción puede mejorar según la estructura y organización del conjunto de datos por analizar. Es decir, factores que ayudan a determinar la complejidad de análisis para un caso de prueba son: tipo de datos, cantidad de variables, datos faltantes, organización y ruido. De lo anterior, se validó que el uso intensivo de procesamiento se ve reducido por este tipo de circunstancia y a diferencia del caso de retinopatía diabética, donde la naturaleza y organización de la información requiere un mayor procesamiento para convertir imágenes en alta resolución en arreglos numéricos para posteriormente alimentar un algoritmo de aprendizaje automático requiere una mayor capacidad de poder computacional y almacenamiento. Esto permite demostrar que el clúster es una herramienta base que cuenta con los componentes necesarios que permiten realizar el análisis de cada caso según sus particularidades y escenarios. Sin embargo, también existen elementos como la habilidad del científico de datos y las estrategias que utiliza para simplificar tareas y disminuir la complejidad del análisis en cuestión. De ese modo, dependiendo de la combinación de herramientas

que este último utilice, en algunos casos los procesos del análisis de datos serán más eficientes que en otros, dependiendo el contexto y particularidad de cada caso de prueba.

Para todos los casos de prueba se validó que utilizar una implementación *standalone*[1] en comparación del PaaS para *Big Data*, conlleva mayor gasto de recursos en infraestructura computacional y complejidad técnica en la configuración de tecnologías y componentes requeridos por implementaciones *Big Data*. Es decir, operar todo desde un modelo de servicio PaaS usando el clúster tiene los beneficios del cómputo de alto rendimiento, recursos elásticos y a la medida, medición y monitoreo de servicios sobre cada implementación, tolerancia a fallas y generación de modelos personalizados de PaaS.

Además, fue validada en el clúster la funcionalidad de aislamiento de recursos (ver Figura 5.4-(1)), este permite que cada usuario pueda reservar recursos de disco duro, procesador y memoria dando la impresión de tener su propio clúster. El monitoreo de cargas de trabajo distribuidas, comprueba que siempre se coordina de manera eficiente la asignación de tareas (ver Figura 5.4-(2)), aprovechando mejor los recursos de infraestructura.

A modo de resumen, se verificó que el clúster es una herramienta que puede integrar los componentes necesarios para dar solución a las particularidades de cada caso prueba y que el desempeño del mismo varia según la selección de componentes que el científico de datos realice en base a su formación y experiencia. Además, en general el desempeño del clúster siempre será de manera distribuida obteniendo ventajas en el uso de infraestructura computacional.

[1] Según https://bit.ly/3yZyKqx es una implementación con sus propios recursos y generalmente no interactúa con otros servicios o aplicaciones

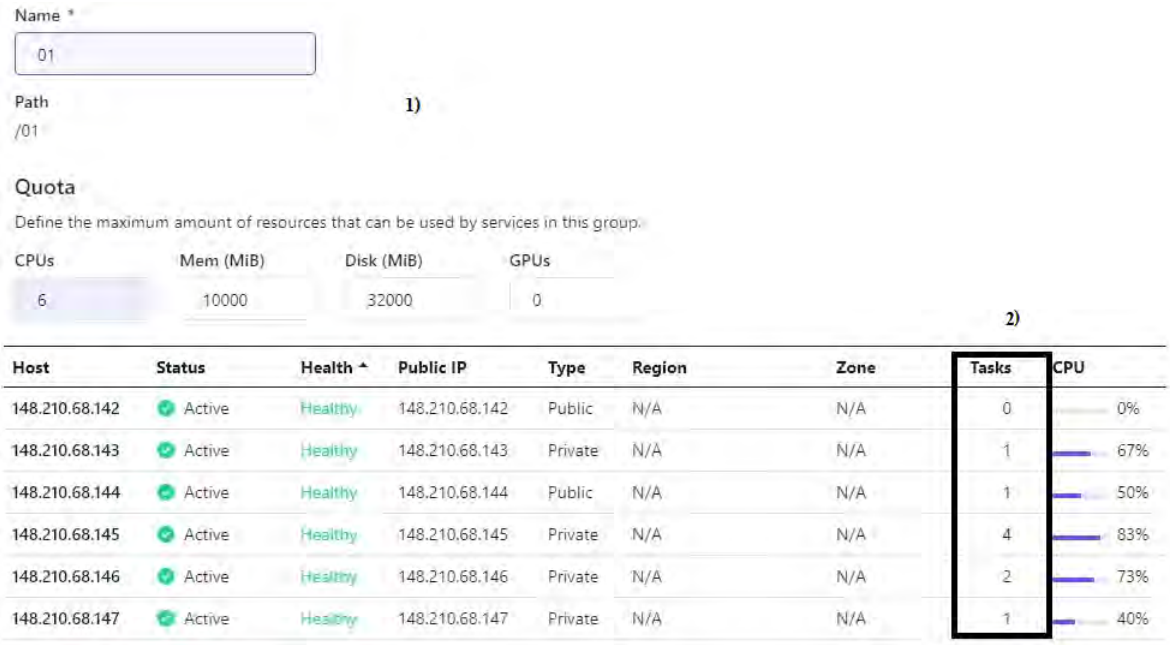


Figura 5.4: Aislamiento de recursos y cargas distribuidas de trabajo.

Capítulo 6

Conclusiones

Con el desarrollo de este proyecto se logró integrar un PaaS con los componentes *Big Data* para la detección de anomalías aprovechando los contextos modernos del cómputo de alto rendimiento, contenedores y redes informáticas de alta velocidad. Del trabajo realizado en esta implementación se concluye lo siguiente:

Se comprobó que una implementación en el clúster bajo el modelo de cómputo distribuido aprovecha al máximo la administración, balance de recursos y asignación de tareas sobre los servidores que están con cargas moderadas o leves de trabajo. Este mismo modelo no se puede replicar usando una implementación *standalone* que utiliza infraestructura como servicio.

La complejidad de configuración se reduce bajo un esquema de contenedores en comparación con una implementación *standalone* en LANTI, facilitando el despliegue servicios y arquitecturas previamente configuradas. Para llegar a esta conclusión, durante la instalación de componentes en las primeras pruebas realizadas del clúster se integró Hadoop, sin previo conocimiento de la configuración en ese momento, habilitando en producción dicho componente con un aproximado de once servidores en un rango de cinco a quince minutos.

Uno de los criterios que influirá drásticamente en el desempeño de una aplicación relacionada con *Big Data* que un ingeniero o científico de datos pueda desarrollar, dependerá de las habilidades, conocimientos y experiencia que pueda tener sobre la utilización de componentes

sin importar que tenga recursos excedentes de infraestructura computacional.

La configuración de un entorno de desarrollo relacionado con la ciencia de datos y pruebas con casos de prueba en detección de anomalías, puede contribuir a la implementación de prototipos y aplicaciones puestos a prueba en un esquema real de producción. Utilizando el clúster de este trabajo se puede acceder al aislamiento de recursos, permitiendo un control de fallas y de interacción entre aplicaciones. Es decir, se puede brindar un espacio de pruebas sin afectar otras implementaciones en producción o de otros usuarios, teniendo como ventaja que ante cualquier desastre no se verán afectados otros servicios o implementaciones en funcionamiento continuo en modo de servicio a usuarios finales.

El cómputo de alto rendimiento y la compatibilidad de contenedores de terceros o propios, convierte el clúster en una herramienta flexible y escalable para la implementación de servicios, aplicaciones y desarrollo de prototipos tanto de prueba como para puesta en producción usando el esquema de redes informáticas de alta velocidad y hardware de bajo costo con flujos continuos de información. De ese modo, este tipo de tecnologías puede contribuir a la investigación poniendo a disposición de alumnos e investigadores.

Apéndice A

Casos de Prueba

A.1. Diagnóstico de retinopatía diabética en imágenes de alta resolución

El diagnóstico clínico de retinopatía diabética DR (por sus siglas en inglés), es una tarea que involucra la asistencia de uno o varios especialistas médicos experimentados en dicha enfermedad. De ese modo, para detectar este padecimiento ocular hay que valorar características muy finas que requieren el apoyo de un sistema de clasificación complejo, dicho lo anterior, esta puede ser una actividad difícil y lenta [80]. En este estudio de caso, se propone un enfoque de redes neuronales convolucionales CNN (por sus siglas en inglés) para diagnosticar DR. De tal modo que, a partir de imágenes a color y con un diagnostico previo, se entrenó una red con arquitectura CNN para un diagnóstico automático y sin intervención del especialista. La muestra se compone por de 62 imágenes que se dividen en dos partes: la primera con valoración negativa y la segunda con retinopatía en nivel proliferativo.

Definición de directorios

En este procesos se definen las librerías necesarias para el funcionamiento de la aplicación, entre ellas: OpenCV para lectura y procesamiento de imágenes; tqdm para utilizar barras

de progreso; Numpy para arreglos numéricos entre otras. Además, se definen dos variables para indicar los directorios que contienen las imágenes de entrenamiento y prueba. En suma, se define el nombre del modelo predictivo con `MODEL_NAME` y la variable LR para definir la unidad de rectificación.

```
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm

TRAIN_DIR='train'
TEST_DIR='test'
IMG_SIZE=500
LR=1e-3

MODEL_NAME='rd--{}--{}.model'.format(LR,'6conv-basic')
```

Etiquetado de las imágenes

Los siguientes comandos simplifican las etiquetas de las imágenes procesadas, utilizando gndr para imágenes cuyo diagnostico es negativo de cualquier signo de retinopatía y prd para casos proliferativos de esta enfermedad.

```
def label_img(img):
    world_label = img.split('.')[-3]
    if world_label== 'gnrd': return[1,0]
    elif world_label == 'prd': return[0,1]
```

Función para procesamiento de imágenes en conjunto de entrenamiento

El siguiente segmento de código define una función auxiliar para cargar en memoria las imágenes de entrenamiento y convertirlas en arreglos numéricos.

```
def create_train_data():
```

```
    training_data=[]
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label=label_img(img)
        path=os.path.join(TRAIN_DIR,img)
        img=cv2.resize(cv2.imread(path,cv2.IMREAD_GRAYSCALE),(IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy',training_data)
    return training_data
```

Función para procesamiento de imágenes en conjunto de prueba

El siguiente segmento de código define una función auxiliar para cargar en memoria las imágenes de prueba y convertirlas en arreglos numéricos. De ese modo, esta actividad es de suma importancia debido a que si no se estandarizan las imágenes se pueden generar ruido y obtener un entrenamiento ineficiente de la red neuronal. Por consiguiente, para este caso particular, se eliminó la saturación de color utilizando escala de grises para el conjunto de entrenamiento y prueba.

```
def process_test_data():
    testing_data=[]
    for img in tqdm(os.listdir(TEST_DIR)):
        path=os.path.join(TEST_DIR,img)
        img_num=img.split('.')[0]
        img=cv2.resize(cv2.imread(path,cv2.IMREAD_GRAYSCALE),(IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img),img_num])
    np.save('test_data.npy',testing_data)
    return testing_data
```

Invocación y carga de imágenes

El siguiente segmento de código invocan las funciones que procesan las imágenes de los conjuntos de entrenamiento.

```
train_data=create_train_data()
#if you already have a train data:
```

```
#train_data=np.load('train_data.npy',allow_pickle=True)
```

Definición de la red neuronal convolucional

La arquitectura de la red neuronal consta es de nueve capas, las capas de convolución se definieron con 32, 64 y 128 entradas y la unidad de aprendizaje rectificado Relu (por sus siglas en inglés). Además, dos entradas iniciales en la primer capa y dos resultados de salida para el diagnostico negativo y proliferativo.

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

import tensorflow as tf
tf.reset_default_graph()


convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 32, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
```

```
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 32, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam',
learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet)
```

Guardar el modelo

Cuando se entrena una red neuronal se guarda el modelo final que ya tiene las caracteristicas funcionales previstas. Dicho proceso es realizado para no repetir el proceso de entrenamiento. El siguiente código se utiliza para cargar un modelo existente.

```
if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model_loaded!')
```

Carga de los datos numéricos

El siguiente código se utiliza para definir dos grupos de imágenes, segmento de entrena-

miento y de prueba.

```
train=train_data[0:20]
```

```
test=train_data[10:]
```

Definición de las entradas de la red neuronal

El siguiente código divide los segmentos de entrenamiento y prueba en las entradas de la red neuronal.

```
X=np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y=[i[1] for i in train]
test_x=np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y=[i[1] for i in test]
```

Entrenamiento de la red neuronal

El siguiente código se utiliza para entrenar la red neuronal pasándole de argumento las variables de entrenamiento y prueba. Además, se definen la cantidad de épocas o iteraciones que ejecutará la red neuronal para su entrenamiento. No hay una cantidad óptima de iteraciones, estas se definen usando una aproximación inicial y se ajusta posteriormente hasta obtener una confianza adecuada del modelo, por ejemplo , un rango del 80% o superior es aceptable relativamente.

```
model.fit({'input': X}, {'targets': Y}, n_epoch=20,
validation_set=({'input': test_x},
{'targets': test_y}),snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

Guardar el modelo de entrenamiento

Cuando el modelo esta entrenado con un grado de confianza aceptable se guarda con el siguiente código.

```
model.save(MODEL_NAME)
```

Predicción

Usando imágenes del conjunto de prueba, se realiza la validación de la red neuronal, para dicha tarea se considera que esta última ya fue previamente entrenada. Además, las imágenes de prueba ya fueron preprocesadas y convertidas en arreglos numéricos, por lo

tanto, esta actividad se enfoca en validar el modelo.

```
import matplotlib.pyplot as plt
#if you dont have this file yet
test_data=process_test_data()
#if you already have it
#test_data=np.load('test_data.npy')
fig=plt.figure()

for num, data in enumerate(test_data[:12]):
    #negativo[1,0]
    #positivo[0,1]

    img_num=data[1]
    img_data=data[0]

    y=fig.add_subplot(3,4,num+1)
    orig=img_data
    data=img_data.reshape(IMG_SIZE,IMG_SIZE,1)

    model_out=model.predict([data])[0]

    if np.argmax(model_out)==1:str_label='N'
    else: str_label='y'
    y.imshow(orig,cmap='gray')
    plt.title(str_label+img_num)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

Resultados

Antes de comentar los resultados finales, es preciso indicar que en previos experimentos y

Figura A.1: Diagnóstico de retinopatía con Red Neuronal Convolucional.

usando la totalidad del conjunto de entrenamiento, se obtuvieron resultados ineficientes en la tarea de clasificación [80]. Por consiguiente, debido a variaciones y ruido en las imágenes, como diferentes niveles de iluminación, ángulos y enfoques del fondo de ojo generan confusión en el entrenamiento de la red neuronal. De ese modo, se puede concluir que la parte de estandarizar las imágenes con atributos comunes a todas las imágenes y la ayuda de un experto médico en el área, podría mejorar el desempeño de la red neuronal. En la Figura A.1 se muestra una clasificación errónea de todas las muestras, en algunas pruebas se obtuvo un máximo del cincuenta por ciento de precisión. Sin embargo, como fue mencionado el ruido de los datos produjo estos resultados.

A.2. Diagnóstico de anomalía en caja de engranajes de turbina eólica

El siguiente estudio de caso está relacionado a la detección de anomalías de una caja de cambios[1] (transmisión) para turbinas eólicas. Para dicha tarea, se utilizó un conjunto de datos el cual registra las vibraciones en diferentes condiciones de carga. De ese modo, para este caso particular se analizó una carga del 90 por ciento, con cuatro sensores colocados en

[1] imagen tomada de: `https://glossarissimo.files.wordpress.com/2017/07/turbina_eolica.jpg?w=1100`

distintos lugares dentro de la caja y a 30 Hz de frecuencia de grabación. Además, las dos variables que representan el estado de la caja son: 0) Condición saludable y 1) Condición de diente roto. Además, cuatro atributos que representan las lecturas de cada sensor registradas eventualmente.

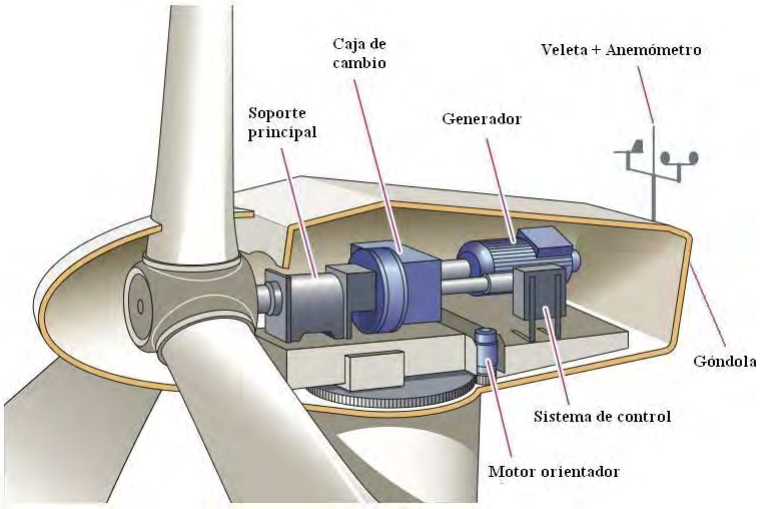


Figura A.2: Turbina éolica.

Importación de biblioteca y carga de datos

En esta etapa inicial, se importan las librerías utilizadas y el conjunto de datos. Además, se definen las columnas que contienen las variables de prueba y predicción. Los siguientes comandos son utilizados en el lenguaje de programación Python y estos son:

```
# first neural network with keras tutorial
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
import matplotlib.pyplot as plt

dataset = loadtxt('transmisionz.csv', delimiter=',')
```

```
# split into input (X) and output (y) variables

X = dataset[:,0:4]
y = dataset[:,4]
```

Arquitectura de la red neuronal

Para definir la red neuronal se utilizó Keras, dicha librería permite utilizar funciones de redes neuronales a través de Tensorflow. Entre las múltiples opciones, de *Frameworks* de redes neuronales, seleccionamos este último por el robusto soporte que ha tenido durante los últimos años.

Primero, se define el modelo secuencial y se agregan las capas de una en una hasta obtener una arquitectura inicial de red, considerada adecuada. Lo primer capa de entrada, debe definirse con el número correcto de variables de entrada, esto es variable, según cada estudio de caso y sus respectivas entradas y salidas.

La primera capa de entrada se define con el parámetro: `input_dim` y se establece el número de cuatro entradas, esto corresponde con las variables de lectura de cada sensor.

Las capas completamente conectadas se definen usando la clase Densa. En dicha clase se pueden especificar el número de neuronas o nodos en cada capa. Como primer argumento, recibe la función de activación de unidad lineal rectificada denominada ReLU en las dos primeras capas y la función Sigmoide en la capa de salida.

La función Sigmoidea se utiliza en la capa de salida para garantizar que los parámetros estén en el rango de valores 0 y 1.

La arquitectura de la red, consta de una capa oculta con 12 nodos, la segunda capa oculta tiene 8 nodos, además función de activación relu.

```
# define the keras model
model = Sequential()
model.add(Dense(12, input_dim=4, activation='relu'))
model.add(Dense(4, activation='relu'))
```

```
model.add(Dense(1, activation='sigmoid'))
```

Compilar el modelo keras

La compilación del modelo requiere tres parámetros: *loss*, se relaciona con el parámetro de perdida del aprendizaje en el ciclo de entrenamiento; *optimizer*, es un optimizador de modelo de red neuronal, lo que acelera el aprendizaje dependiendo de uno u otro. El parámetro *metrics*, es utilizado como métrica de medición en la confianza del modelo de red neuronal, entre mas alto mejor, significa un modelo y sus predicciones son confiables. La compilación se ejecuta con el siguiente comando:

```
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Ajuste del modelo

```
# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
```

Evaluación del modelo

```
# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100)))
```

Predicción

Una vez que el modelo está entrenado, se utilizaron los primeros cinco registros del conjunto de datos para entrenar la red, estos datos corresponden con los valores que se registraron de los sensores cuando no existe falla en la caja de la transmisión de la turbina eólica. En este caso la carga es de 90 y la frecuencia de 30Hz. Los siguientes comandos ejecutan la iteración para hacer la predicción de los de los cinco registros, que en teoría deben dar como resultado cero. La Figura A.3 muestra los resultados de salida.

```
predictions = model.predict_classes(X)
# summarize the first 5 cases
```

```
for i in range(5):
print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i])
```

```
Confianza: 96.30
[5.4304, -0.0607, -4.7702, 0.5993] => 0 (Expectativa 0)
[-15.4611, 1.9086, 2.5493, -0.4795] => 0 (Expectativa 0)
[-5.3769, -1.3211, 3.0183, -0.3606] => 0 (Expectativa 0)
[8.8753, 3.3958, -1.8584, 2.4127] => 0 (Expectativa 0)
[3.1497, 2.3908, 1.1288, 3.2472] => 0 (Expectativa 0)
```

Figura A.3: Resultados de la predicción.

Épocas

Son iteraciones realizadas durante el entrenamiento de una red neuronal. Por consiguiente, la cantidad de épocas es definida por el usuario y no siempre se logra un entrenamiento adecuado con el número establecido. Funcionan como una guía para monitorear la precisión obtenida en el entrenamiento del modelo de red neuronal. Para este caso particular, se obtuvo un entrenamiento completo al 100%, en la Figura A.4 muestra que el entrenamiento terminó en la época 150 aproximadamente.

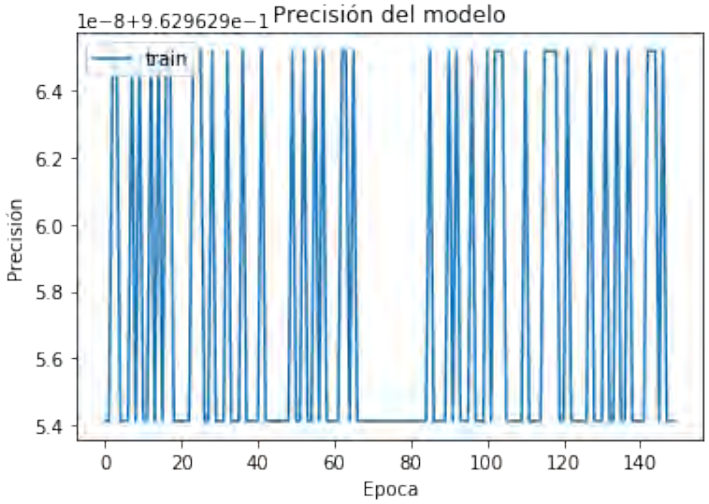


Figura A.4: Épocas del entrenamiento.

Perdidas

En redes neuronales, existe un término que se le denomina perdida, este le resta valor a la confianza del modelo durante el ciclo de entrenamiento, al inicio esta perdida es grande, conforme avanzan las épocas estas perdidas cada son menores, como se muestra en la Figura A.5 una curva registra las perdidas durante los ciclos de entrenamiento.

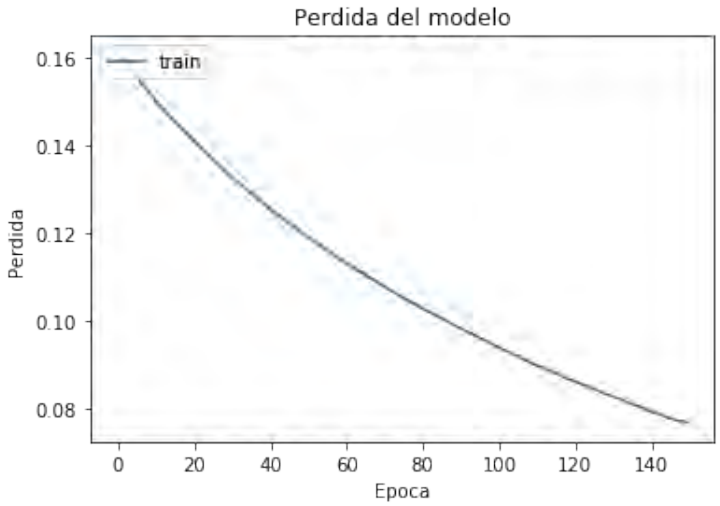


Figura A.5: Gráfico de perdida.

A.3. Caso de prueba para detección de anomalías en placas de acero

Para este caso de prueba se utilizó un conjunto de datos para predecir anomalías en la manufactura de placas de acero. La clasificación consiste en siete tipos distintos de fallas que afectan la calidad del producto final. El objetivo es utilizar un algoritmo de *Machine Learning*, específicamente un árbol de decisión, para el reconocimiento automático de patrones y saber que tipo de falla tendrá la placa de acero cuando existan una variación anómala en los parámetros del proceso de manufactura.

Descripción de los datos

El conjunto de datos analizado consta de 1,941 registros, y 28 campos, de los cuales tres son categóricos y 25 numéricos.

Tipos de anomalía en las placas de acero

Los tipos de falla en las placas de acero identificados son: golpes, suciedad, calidad, rasguño-Z, rasguño-K, manchas, otras fallas.

A.3.1. Caso de prueba con árboles de decisión

Las siguiente secciones son código de python que se ejecutaron en el cluster de *Big Data*.

Importación de librerías

En este apartado se define la importación de librerías necesarias para usar los árboles de decisión, conjuntos de dato y arreglos númericos.

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

Función para dividir el conjunto de datos

para entrenamiento y prueba se dividen los datos en un 70% para entrenamiento y 20% para validación.

```
def splitdataset(balance_data):

# Seperating the target variable
X = balance_data.values[1:940, 0:26]
Y = balance_data.values[1:940, 27]

# Spliting the dataset into train and test
```

```
X_train, X_test, y_train, y_test = train_test_split(
X, Y, test_size = 0.3, random_state = 100)

return X, Y, X_train, X_test, y_train, y_test
```

Definición del clasificador

El clasificador con los criterios que usa un árbol de decisión, se definen con el siguiente código:

```
    def train_using_gini(X_train, X_test, y_train):



clf_gini = DecisionTreeClassifier(criterion = "gini",
random_state = 100,max_depth=3, min_samples_leaf=5)

# Performing training
clf_gini.fit(X_train, y_train)
return clf_gini
```

Función con entropía

En esta sección se define la función de entrenamiento con entropía (en el modelo de predicción representa la incertidumbre).

```
def tarin_using_entropy(X_train, X_test, y_train):

# Decision tree with entropy
clf_entropy = DecisionTreeClassifier(
criterion = "entropy", random_state = 100,
max_depth = 3, min_samples_leaf = 5)

# Performing training
```

```
clf_entropy.fit(X_train, y_train)
return clf_entropy
```

Función de predicción

En esta sección se define la función de predicción y se alimenta con los datos del conjunto.

```
   def prediction(X_test, clf_object):

# Predicton on test with giniIndex
y_pred = clf_object.predict(X_test)
print("Predicted values:")
print(y_pred)
return y_pred
```

Precisión del modelo

Este apartado imprime el porcentaje de precisión del modelo de predicción.

```
  # Function to calculate accuracy
   def cal_accuracy(y_test, y_pred):

print("Matriz de Confusion: ",
confusion_matrix(y_test, y_pred))

print ("Precisión: ",
accuracy_score(y_test,y_pred)*100)

print("Report : ",
classification_report(y_test, y_pred))
```

Función principal

En este modulo se invocan los módulos necesarios para ejecutar de una sola vez el algoritmo árboles de precisión.

```
def main():

# Building Phase
data = importdata()
X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
clf_gini = train_using_gini(X_train, X_test, y_train)
clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

# Operational Phase
print("Results Using Gini Index:")

# Prediction using gini
y_pred_gini = prediction(X_test, clf_gini)
cal_accuracy(y_test, y_pred_gini)

print("Results Using Entropy:")
# Prediction using entropy
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)



# Llamando la función
if __name__=="__main__":
main()
```

Predicción

La siguiente predicción fue llevada a cabo con entropía obteniendo las mejores predicciones de fallas en las placas de acero en suciedad (Dirtiness) y manchas (Stains), arañazo

en Z (ZScratch) como se puede ver en la Figura A.6, con una muestra de 940 registros.

```
Accuracy :  82.62411347517731
Report :                precision    recall  f1-score   support

       Bumps       0.00      0.00      0.00        16
   Dirtiness       1.00      0.28      0.43        18
    K_Scatch       0.98      0.97      0.98       119
      Pastry       0.57      0.77      0.65        48
      Stains       1.00      0.72      0.84        18
   Z_Scratch       0.77      0.98      0.86        63

    accuracy                           0.83       282
   macro avg       0.72      0.62      0.63       282
weighted avg       0.81      0.83      0.80       282
```

Figura A.6: Resultados de la predicción de árbol.

A.3.2. Caso de prueba con algoritmo kmeans y spark

Este algoritmo se utilizo en el Capítulo 3, en la Etapa 3.6. Sin embargo, para este caso de prueba, un nuevo conjunto de datos es utilizado relacionado con defectos en placas de acero. El objetivo es clasificar las fallas en grupos en relación a los parámetros de entrada.

Importación de librerias

Para el adecuado funcionamiento de una aplicación Spark se debe incluir un conjunto de bibliotecas que proporciona el *framework*. Por ejemplo: kMeans, con esta libreria se manda llamar implementación del algoritmo del mismo nombre; SparkContext, permite crear un contexto de trabajo de la aplicación; Matplot, utilizada para gráficos bidimensionales; VectorAssembler, permite generar vectores de datos. los siguientes comandos muestran las bibliotecas que se utilizan en la aplicación.

```
from pyspark.ml.clustering import KMeans
#sc = SparkContext("local", "First App")
from pyspark.sql import SparkSession
```

```
import matplotlib.pyplot as plt
from pyspark.ml.feature import VectorAssembler
```

Carga de datos desde Hadoop y definición de contexto

Los siguientes comandos tienen por objetivo definir la sesión Spark y cargar los datos desde el directorio de Hadoop.

```
spark = SparkSession.builder.appName('x').getOrCreate()
df=spark.read.csv('steel_plate_faults_PX.csv',
mode="DROPMALFORMED",inferSchema=True, header = False) #df.show())
```

Ensamblar el vector de datos

En este paso se deben definir los atributos de cada cada placa de acero y son guardado como vectores.

```
vecAssembler = VectorAssembler(inputCols=["_c0", "_c1","_c2","_c3","_c4","_c5","_c6",
"_c7","_c8","_c9","_c10","_c11","_c12","_c13",
"_c14","_c15","_c16","_c17","_c18","_c19",
"_c20","_c21","_c22","_c23","_c24","_c25","_c26","_c27" ], outputCol="features")
new_df = vecAssembler.transform(df)
#new_df.show()
```

Entrenamiento del modelo

Antes de entrenar el modelo, se defienen siete clústers para clasificar las fallas, es decir cada conjunto de atributos será clasificado en uno de estos segmentos. Los siguientes comandos mostrados a continuación se utilizan para entrenar el modelo con el algoritmo kmeans de *Machine Learning*.

```
kmeans = KMeans(k=7, seed=1)  # 7 clusters here
model = kmeans.fit(new_df.select('features'))
```

Predicción

El resultado de la predicción es inmediato, cada registro es agrupado a un clúster. Es decir, según los atributos de cada registro se clasifica en un grupo que debe corresponder con una falla en particular.

```
transformed = model.transform(new_df).select("features", "prediction","_c27")
#transformed.describe().show()
#transformed.show(300)

cl_2 = transformed.filter(transformed.prediction==6)
#cl_2.show(transformed.count())
cl_2.show(transformed.count())
```

Discusión caso de placas de acero

En este caso de prueba, la clasificación de anomalías en placas de acero no concuerda con los resultados que se obtuvieron con la predicción. Para confirmar esto primero se analiza cual es la media o el grupo de falla mas común, según la predicción (prediction, como aparece en la Figura A.7), muestra que la media se encuentra en el rango del grupo tres y cuatro.

Esto quiere decir que lo mas normal es que la placa esta rayada o sucia, y según la clasificación que hizo la industria, la media se localiza en los rayones tipo Z y K (se desconoce porque se nombraron así) en las placas. Otra prueba de la mala clasificación de la industria es la siguiente: el resultado del grupo cero en la predicción abarca mas registros en comparación con la cantidad que la industria clasificó en esa categoría. La Figura A.8 y A.9 muestran el grupo de falla y la predicción obtenida con el algoritmo kmeans.

En síntesis, lo que se debe hacer es replantear el proceso de verificación de fallas, debido a que una inspección visual podría ser un error para ello al existir atributos latentes no detectadas en ese proceso de verificación a través de la observación.

summary	prediction	_c27
count	1941	1941
mean	2.7990726429675425	4.841318907779495
stddev	2.054369034340315	2.1441753062941697
min	0	1
max	6	7

Figura A.7: Descripción estadística de las predicciones.

features	prediction	c27
[645.0,651.0,2538...	0	1
[1363.0,1372.0,21...	0	1
[1358.0,1372.0,23...	0	1
[830.0,837.0,1874...	0	1
[91.0,104.0,23781...	0	1
[161.0,168.0,2891...	0	1
[1328.0,1338.0,26...	0	1
[1259.0,1271.0,19...	0	1
[1140.0,1147.0,23...	0	1
[13.0,24.0,251289...	0	1
[11.0,20.0,284522...	0	1

Figura A.8: Clasificación del cluster cero.

[1027.0,1034.0,29...	0	1
[1465.0,1486.0,18...	0	1
[1360.0,1373.0,20...	0	1
[1155.0,1167.0,21...	0	1
[1325.0,1334.0,19...	0	1
[451.0,465.0,2550...	0	1
[30.0,38.0,227767...	0	1
[1166.0,1185.0,22...	0	2
[173.0,188.0,1835...	0	2
[83.0,93.0,184628...	0	2
[17.0,27.0,185961...	0	2
[98.0,108.0,18894...	0	2
[39.0,215.0,30033...	0	3
[836.0,878.0,2150...	0	3
[414.0,438.0,2887...	0	3
[1213.0,1219.0,19...	0	4
[1302.0,1308.0,23...	0	5
[1315.0,1321.0,23...	0	5

Figura A.9: Clasificación del cluster cero segunda parte.

Apéndice B

Configuraciónes

B.1. Parametrización de Nodos

La funcionalidad de cada nodo fue descrita en El Capítulo 2, en la sección 2.2. En la Figura B.1 se ilustra la actividad llevada a cabo de configuración de servicios particularizada para cada nodo.

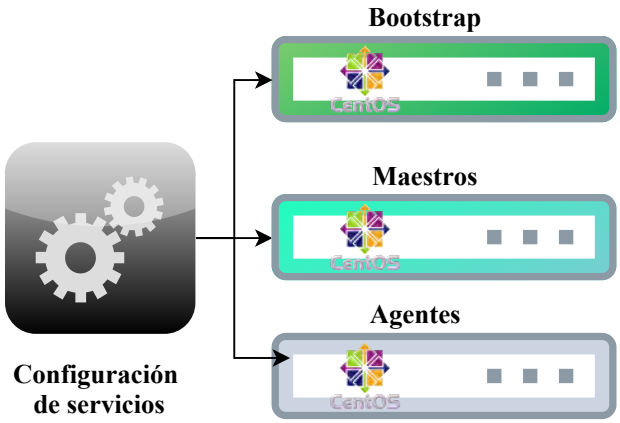


Figura B.1: Configuración de servicios.

1. Configuración nodo *bootstrap* primera parte

En este paso es necesario generar las claves publicas requeridas por el protocolo ssh. Este último es utilizado para establecer conexiones remotas por terminal utilizando el algoritmo de cifrado RSA. Con el primer comando, mostrado al final de este párrafo, se realiza dicha tarea y el resto de las lineas de código establece una copia a cada uno de los nodos del clúster.

```
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.141 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.142 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.143 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.144 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.145 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.146 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.147 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub |
ssh root@148.210.68.148 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

La instalación del sistema DCOS exige un preconfiguración en el SO de cada uno de los equipos de cómputo que conforman el clúster. Entre estos requisitos tenemos:

2. Autorización de usuario root para ejecución remota de comandos

Para ejecutar comandos de forma remota sobre el sistema operativo CentOs a usando el protocolo ssh, es necesario contar con un usuario dado de alta en en cada nodo con permisos asignados al grupo al que pertenece, o asignar al administrador del sistema para dicha tarea. Esta última configuración fue utilizada en la implementación del clúster. Sin

embargo, con el fin de establecer un protocolo adecuado de seguridad, no se debería utilizar el usuario *root* como predeterminado. Para el uso de pruebas, se optó activar el esquema descrito mediante la edición del archivo de configuración localizado en la siguiente ruta:`/etc/sudoers`

```
#Se descomenta la siguiente línea, borrando el símbolo (#)
#% root ALL = (ALL) ALL
#El resultado final es:
% root ALL = (ALL) ALL
```

3. Sincronización de horario

El protocolo de horario de red (NTP, por sus siglas en inglés), se debe activar en todos los nodos del clúster para la sincronización del reloj. De manera predeterminada, durante el inicio de DCOS, recibirá un error si no está habilitado. Para verificar si este servicio ya está en funcionamiento ejecutar el siguiente comando:

```
ntptime adjtimex -p timedatectl
```

El comando anterior ajusta la fecha y hora con un servidor de tiempo. Se puede verificar verificar el ajuste de fecha, si usamos el sig. comando: `datetime`

4. Distribución de llaves públicas

Como ya se mencionó, el algoritmo de cifrado RSA requiere que cada nodo distribuya su llave publica a todos los nodos integrantes del clúster, misma que servirá durante la solicitud de conexión entre ellos, a través del protocolo ssh. Para lograr este objetivo es necesario ejecutar el siguiente comando:

```
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.141 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.142 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.143 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

```
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.144 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.145 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.146 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.147 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
cat ~/.ssh/id_rsa.pub
| ssh CentOs@148.210.68.148 "mkdir ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

5. Configuración de repositorio Docker

Cada nodo debe tener acceso a un repositorio Docker público o a un registro interno. Esto con el fin de activar un servicio para el uso de contenedores Docker. Como ya se mencionó, DCOS opera con un servicio basado en contenedores por lo que es crucial activar este servicio. Para añadirlo el repositorio se debe ejecutar el siguiente comando como administrador del sistema:

```
sudo tee /etc/yum.repos.d/docker.repo <<-EOF
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/CentOs/$releasever/
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF
```

6. Configuración de cortafuegos

Existe un problema conocido en el que el proceso del *firewall* interactúa de manera deficiente Docker. Lo recomendado, según [61], es detenerlo y desactivarlo, para eso se

debe ejecutar el siguiente comando:

```
sudo systemctl stop firewalld && sudo systemctl disable firewalld
```

7. Habilitar OverlayFS

Es una implementación para Linux, consiste unir un sistema de archivos o varios en uno solo. Para habilitar este componente se utiliza el siguiente comando:

```
sudo tee /etc/modules-load.d/overlay.conf <<EOF
overlay
EOF
```

8. Configuración de módulo de seguridad SELINUX

Este módulo permite mantener en funcionamiento reglas de seguridad; sin embargo, como ya se explicó anteriormente, por motivos académicos y de pruebas en este clúster, la configuración de seguridad para este módulo se desactivó. El comando ejecutado es el siguiente:

```
sudo sed -i s/SELINUX=enforcing/SELINUX=permissive/g /etc/selinux/config &&
```

9. Añadir grupos

La configuración previa de instalación DCOS, según [61], recomienda agregar dos nuevos grupos de usuarios, en cada uno de los nodos integrantes del clúster. Para ello se ejecutaron los siguientes comandos:

```
sudo groupadd nogroup &&
sudo groupadd docker
```

10. Recargar módulos del kernel

Para dicha tarea es necesario reiniciar el sistema operativo ejecutando el siguiente comando: `reboot`

11. Establecer el idioma

Para establecer el idioma se utilizó el siguiente comando:

```
localectl set-locale LANG=en_US.utf8
```

esto con el fin de mantener una configuración uniforme en las variables de entorno de cada uno de los nodos.

12. Configuración DNSmasq

Este actividad consiste en detener el servicio DNSmasq, de ese modo, se asegura que, durante el proceso de instalación DCOS no se presenten problemas con la comunicación de red puerto 53. El comando para dicha tarea es:

```
sudo systemctl stop dnsmasq && sudo systemctl disable dnsmasq
```

13. Instalación de paquetes de compresión de datos

Algunos procesos involucrados con el objetivo de instalación del cluster requieren algunas utilidades de compresión de datos, como: UnZip, GNU tar y XZ Utils instaladas en los nodos del clúster. Con el siguiente comando se instalan las utilerias mencionadas:

```
yum install -y nano ntp tar xz unzip
curl ipset open-vm-tools nfs-utils yum-versionlock
chkconfig ntpd on
service ntpd restart
systemctl enable ntpd
yum -y update
```

14. Instalación de Docker

Los siguientes comandos tienen como finalidad instalar el motor de contenedores Docker, para ello en una terminal ejecutar:

```
sudo yum install -y docker-engine-1.13.1
docker-engine-selinux-1.13.1
yum versionlock docker-engine docker-engine-selinux
sudo systemctl start docker
sudo systemctl enable docker
```

15. Configuración del proceso residente systemd

para iniciar el servicio de gestión de contenedores (Docker Daemon, por su nomenclatura en inglés) con OverlayFS, ejecutar el siguiente comando:

```
sudo mkdir -p /etc/systemd/system/docker.service.d && sudo tee
/etc/systemd/system/docker.service.d/override.conf <<-EOF
[Service]ExecStart=ExecStart=/usr/bin/dockerd
--storage-driver=overlay
EOF
```

16. Pruebas con Docker

Los siguientes comandos tienen por objetivo validar la instalación de la herramienta Docker, desde una terminal se ejecutan:

```
sudo docker ps
docker run hello-world
```

El comando: `docker run` imprime texto en pantalla "hello-world" y es indicativo que la tecnologia Docker esta preparada para ser utilizada.

17. Configuración Bootstrap segunda parte

Para instalar Docker NGINX (un servidor web de alto rendimiento), ejecutamos los siguientes comandos:

```
docker pull nginx:alpine docker run -d --restart=unless-stopped -p 8081:80 -v
/opt/dcos-setup/genconf/serve: /usr/share/nginx/html:ro --name=dcos
```

18. Implementar un script de detección de IP

En este paso, es necesario generar un *script* para detección de IP de los nodos que conformaran el clúster el cual tiene las siguientes funcionalidades y consideraciones:

- Cada nodo en un clúster DCOS tiene una dirección IP única que se utiliza para comunicarse entre los nodos en el clúster.
- Imprimir la dirección IPv4 única de un nodo en *STDOUT* cada vez que se inicia DCOS en el nodo.

- La dirección IP de un nodo no debe cambiar después de que DCOS esté instalado en el nodo. Si la dirección IP de un nodo cambia, el nodo debe desinstalarse.
- El *script* debe devolver la misma dirección IP que la especificada en config.yaml.
- El *script* se debe guardar en el directorio genconf/ip-detect y codificado en UTF-8.

El código del *script* se muestra a continuación:

```
#!/usr/bin/env bash
set -o nounset -o errexit
export PATH=/usr/sbin:/usr/bin:$PATH
echo $(ip addr show ens160 |
grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'|head -1)
```

Apéndice C

Infraestructura LANTI

C.1. Descripción

El Laboratorio Nacional en Tecnologías de la Información en Ciudad Juárez cuenta con dos clústeres principales. Uno de ellos dedicado al almacenamiento y el segundo al cómputo. El primero es utilizado para ofrecer servicios de virtualización mediante VMware VSphere 6.0. De tal modo, los recursos disponibles de este clúster son de 168 CPU (Unidad Central de Procesamiento, por sus siglas en inglés) son virtuales y 576 GB de RAM. Además, los recursos de esta infraestructura están repartidos en entre dos laboratorios, el primero implementa Ovirt, una tecnología de virtualización distribuida de codigo abierto, para la administración de infraestructuras computacionales. El segundo, es usado para algunas clases de la maestría en cómputo aplicado. También se dispone de un clúster de almacenamiento de aproximadamente 21 TB mediante la implementación de PetaSAN. para el uso de todas las demás implementaciones se utiliza una interfaz iSCSI. La infraestructura se muestra en el diagrama de la Figura C.1.

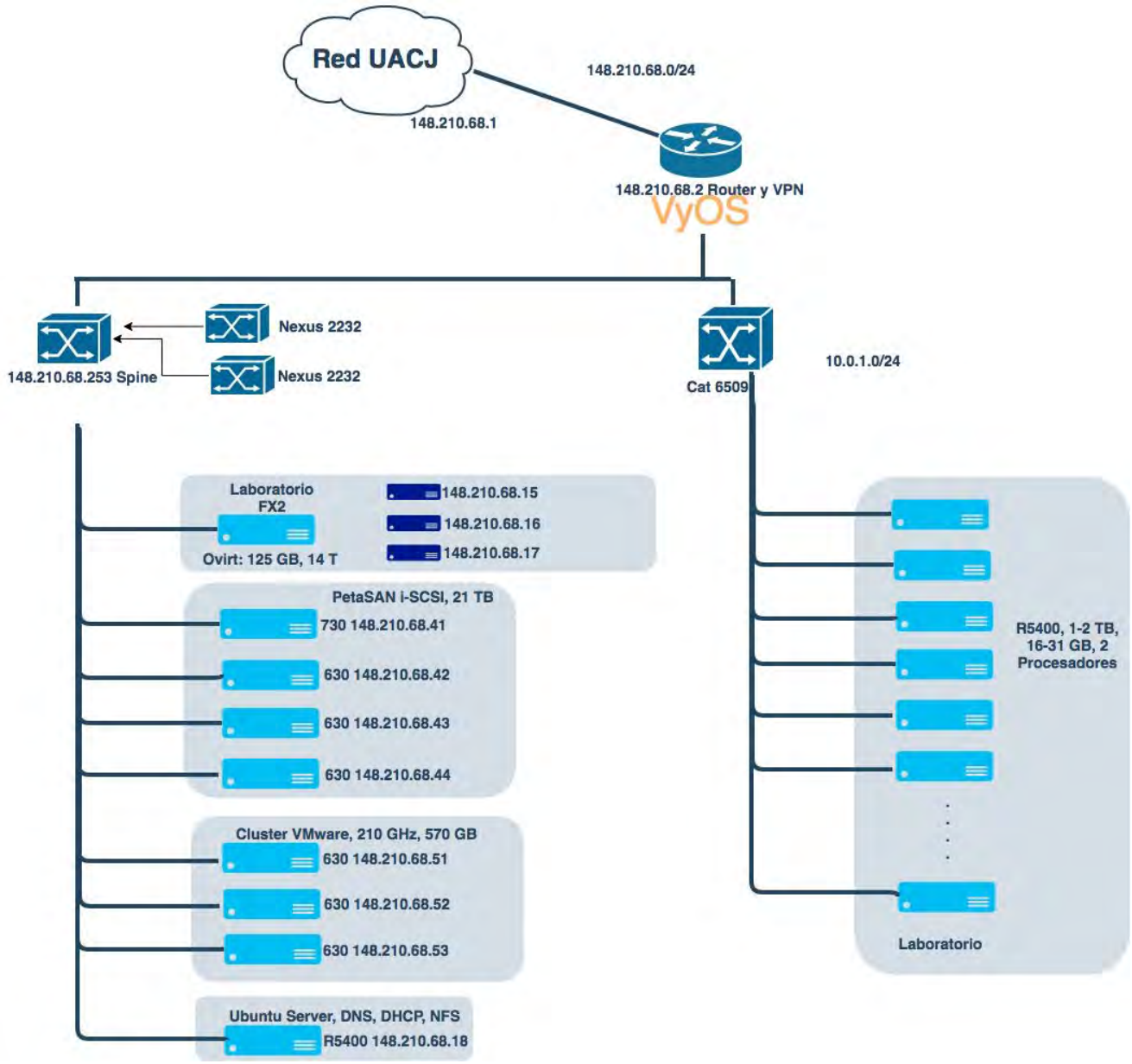


Figura C.1: Diagrama de infraestructura LANTI.

Bibliografía

[1] S. Baller, S. Dutta, B. Lanvin, and others, *Global information technology report 2016*. Ouranos Geneva, 2016.

[2] C. Matt, T. Hess, and A. Benlian, "Digital Transformation Strategies," *Business and Information Systems Engineering*, vol. 57, no. 5, pp. 339–343, 2015.

[3] D. S. Linthicum, "Moving to autonomous and self-migrating containers for cloud applications," *IEEE Cloud Computing*, vol. 3, no. 6, pp. 6–9, 2016.

[4] C. Pahl, "Containerization and the PaaS Cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[5] Z. Li, Y. Zhang, and Y. Liu, "Towards a full-stack devops environment (platform-as-a-service) for cloud-hosted applications," *Tsinghua Science and Technology*, vol. 22, no. 1, pp. 1–9, 2017.

[6] K. Schwertner, "Digital transformation of business," *Trakia Journal of Science*, vol. 15, no. Suppl.1, pp. 388–393, 2017.

[7] J. Akoka, I. Comyn-Wattiau, and N. Laoufi, "Research on Big Data – A systematic mapping study," *Computer Standards and Interfaces*, vol. 54, pp. 105–115, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.csi.2017.01.004

[8] M. Kolanovic and R. Krishnamachari, "Big data and AI strategies: Machine learning and alternative data approach to investing," J.P. Morgan, Tech. Rep. May, 2017.

[9] L. Keighley, *Big Data MBA - Driving Business Strategies with Data Science Bill Schmarzo*. Wiley, 2017, vol. 59.

[10] M. Vozábal, "Department of Computer Science and Engineering Master Thesis Tools and Methods for Big Data Analysis," Ph.D. dissertation, University of West Bohemia, 2016.

[11] K. Weins, "Cloud Computing Trends 2020 State of the Cloud Report | Flexera Blog," 2020. [Online]. Available: https://www.flexera.com/blog/industry-trends/2020/05/trend-of-cloud-computing-2020/

[12] B. Hindman, A. Konwinski, and M. Zaharia, "Mesos: A platform for fine-grained resource sharing in the data center," *Proceedings of NSDI 2011: 8th USENIX Symposium on Networked Systems Design and Implementation*, pp. 295–308, 2011.

[13] Tecnológico Nacional De Mexico, "Laboratorio Nacional De Tecnologías De La Información," pp. 1–1, 10 2020. [Online]. Available: http://lanti.org.mx/lanti/index.php/home/about-us-2

[14] K. Hurst, *Engineering design principles*. Butterworth-Heinemann, 1999.

[15] P. Bourque and R. Fairley, *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, 2014.

[16] T. Limoncelli, S. R. Chalup, and C. J. Hogan, *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems*. Pearson Education, 2014, vol. 2.

[17] B. Acek, K. Ecker, B. Plateau, and D. Trystram, *Handbook On Paralell And Distributed Processing*. Springer Berlin Heidelberg, 2000. [Online]. Available: http://download.springer.com/static/pdf/185/bfm%253A978-3-642-56668-4%252F1.pdf?auth66=1401995374_4d363c462609ae19e3de18b0ee51ef07&ext=.pdf

[18] L. Smarr and C. E. Catlett, "Metacomputing," *Communications of the ACM*, vol. 35, no. 6, pp. 44–52, 1992.

[19] R. Trobec, B. Slivnik, P. Bulić, and B. Robič, *Introduction to parallel computing: from algorithms to programming on state-of-the-art platforms*. Springer, 2018.

[20] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *Special Publication 800-145*, vol. 728, pp. 269–274, 2011.

[21] I. Docker, "Definición de contenedores Docker," 2020. [Online]. Available: https://www.docker.com/resources/what-container

[22] D. S. Terzi, R. Terzi, and S. Sagiroglu, "A survey on security and privacy issues in big data," *2015 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015*, pp. 202–207, 2016.

[23] B. Matturdi, X. Zhou, S. Li, and F. Lin, "Big Data security and privacy: A review," pp. 135–145, 2014.

[24] M. Barika, S. Garg, A. Y. Zomaya, L. Wang, A. V. Moorsel, and R. Ranjan, "Orchestrating Big Data Analysis Workflows in the Cloud," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–41, 2019.

[25] P. Chandarana and M. Vijayalakshmi, "Big data analytics frameworks," *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications, CSCITA 2014*, pp. 430–434, 2014.

[26] M. Angélica, C. Gutiérrez, A. R. Ríos, C. Calero, E. Fernández-Medina, and M. Piattini, "Análisis y revisión de la literatura en el contexto de proyectos de fin de carrera: Una propuesta," *Sociedad Chilena de Ciencia de la Computación*, vol. 6, 2015. [Online]. Available: https://users.dcc.uchile.cl/~mmarin/revista-sccc/sccc-web/Vol6/CCESC08.pdf

[27] A. Dev Mishra and Y. Beer Singh, "Big data analytics for security and privacy challenges," in *IEEE International Conference on Computing Communication and Automation, ICCCA 2016*. IEEE, 2017, pp. 50–53.

[28] V. N. Inukollu, S. Arsi, and S. Rao Ravuri, "Security Issues Associated with Big Data in Cloud Computing," *International Journal of Network Security & Its Applications*, vol. 6, no. 3, pp. 45–56, 2014.

[29] PromptCloud, "Big Data Industry Report," PromtCloud, Tech. Rep., 2016.

[30] Statista, "Volumen total de datos en el mundo 2010-2024," 2020. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

[31] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data Mining and Analytics in the Process Industry: The Role of Machine Learning," *IEEE Access*, vol. 5, pp. 20 590–20 616, 2017.

[32] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[33] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[34] A. Sverdlov, "An overview of machine learning and pattern recognition," *Retrieved June*, vol. 26, p. 2015, 2015.

[35] C. M. Bishop, *Pattern Recognition and Machine Learning*, M. Jordan and J. Kleinberg, Eds. Cambridge: Springer, 2006.

[36] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[37] D. Sullivan, "How Machine Learning Works, As Explained By Google," 2015. [Online]. Available: https://martechtoday.com/how-machine-learning-works-150366

[38] J. Dean, *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. John Wiley & Sons, 2014.

[39] E. Bisong, *Building machine learning and deep learning models on Google cloud platform*. Springer, 2019.

[40] N. J. Nilsson, "Introduction To Machine Learning An Early Draft Of A Proposed Textbook," Department of Computer Science, Tech. Rep. 2, 2005. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21172442

[41] "What Is Machine Learning? | How It Works, Techniques & Applications - MATLAB & Simulink." [Online]. Available: https://www.mathworks.com/discovery/machine-learning.html#how-it-works

[42] E. Alpaydin, "Introduction to Machine Learning Ethem Alpaydin," *Introd. to Mach. Learn*, 2014.

[43] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[44] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[45] S. L. Salzberg, "C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," 1994.

[46] V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman, "Decision trees: an overview and their use in medicine," *Journal of medical systems*, vol. 26, no. 5, pp. 445–463, 2002.

[47] A. E. Mohamed, "Comparative Study of Four Supervised Machine Learning Techniques for Classification," *International Journal of Applied Science and Technology*, vol. 7, no. 2, pp. 5–18, 2017.

[48] O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*. Springer, 2005.

[49] A. Krenker, J. Bešter, and A. Kos, "Introduction to the artificial neural networks," *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pp. 1–18, 2011.

[50] S. S. Haykin and others, "Neural networks and learning machines/Simon Haykin." 2009.

[51] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[52] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[53] Z. Lv, X. Yan, and Q. Jiang, "Batch process monitoring based on just-in-time learning and multiple-subspace principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 137, pp. 128–139, 10 2014.

[54] C. I. Wu, H. Y. Kung, C. H. Chen, and L. C. Kuo, "An intelligent slope disaster prediction and monitoring system based on WSN and ANP," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4554–4562, 8 2014.

[55] V. Ankam, *Big data analytics*. Packt Publishing Ltd, 2016.

[56] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[57] R. Estrada and I. Ruiz, *Big Data SMACK*. Apress, 2016.

[58] D. Kakadia, *Apache Mesos Essentials*, 2015th ed. Birmingham: Packt Publishing Ltd., 2015. [Online]. Available: www.packtpub.com

[59] M. Arrington, "Análisis de plataforma BigML," pp. 1–1, 2020. [Online]. Available: https://www.crunchbase.com/organization/bigml#section-overview

[60] Red Hat, "Red Hat OpenShift," 2021. [Online]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift

[61] Mesosphere, "Documentación oficial DC/OS 2.0," 2020. [Online]. Available: https://docs.d2iq.com/mesosphere/dcos/2.0/

[62] M. Frampton, *Complete Guide to Open Source Big Data Stack*. Apress, 2018.

[63] M. Turck, "Big Data Landscape 2017," p. 1, 2017. [Online]. Available: http://mattturck.com/matt-turck-firstmark-2017-big-data-landscape/

[64] R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," *IEEE Cloud Computing*, vol. 1, no. 1, pp. 78–83, 5 2014.

[65] P. Hunt, M. Konar, Y. Grid, F. Junqueira, B. Reed, and Y. Research, "ZooKeeper: Wait-free Coordination for Internet-scale Systems," *ATC. USENIX*, vol. 8, 2010.

[66] R. Menon, *Cloudera administration handbook*. Packt Publishing Ltd, 2014.

[67] T. White, *Hadoop: The Definitive Guide, Fourth Edition*. O'Reilly Media, Inc, 2015. [Online]. Available: hadoopbook.com

[68] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003, pp. 20–43.

[69] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 1 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1327452.1327492

[70] H. Karau, A. Konwinski, W. Patrick, and Z. Matei, *Learning Spark Lightning-Fast Big Data Analysis*. O'Reilly Media, Inc, 2015, vol. 274.

[71] Y. Huai, A. Chauhan, A. Gates, G. Hagleitner, E. N. Hanson, O. O'Malley, J. Pandey, Y. Yuan, R. Lee, and X. Zhang, "Major technical advancements in apache hive," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1235–1246.

[72] J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, and others, "Achieving 100,000,000 database inserts per second using accumulo and d4m," in *2014 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2014, pp. 1–6.

[73] L. George, *HBase: the definitive guide: random access to your planet-size data*. ."O'Reilly Media, Inc.", 2011.

[74] L. Chang, Z. Wang, T. Ma, L. Jian, L. Ma, A. Goldshuv, L. Lonergan, J. Cohen, C. Welton, G. Sherry, and others, "HAWQ: a massively parallel processing SQL engine in hadoop," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1223–1234.

[75] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and

R. Murthy, "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[76] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1099–1110.

[77] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia, "Spark SQL: Relational Data Processing in Spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1383–1394. [Online]. Available: https://doi.org/10.1145/2723372.2742797

[78] M. F. Sanner, "Python: A programming language for software integration and development," *Journal of Molecular Graphics and Modelling*, vol. 17, no. 1, pp. 57–61, 1999.

[79] P. Romero-Aroca and R. Sagarra Álamo, "La retinopatía diabética e hipertensiva," *AMF*, vol. 14, no. 7, pp. 382–384, 2018.

[80] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, and Y. Zheng, "Convolutional Neural Networks for Diabetic Retinopathy," *Procedia Computer Science*, vol. 90, no. July, pp. 200–205, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2016.07.014