



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal

Ambiente de análisis de datos mediante Big Data
(Luminus)

2018-A104

Que para cumplir con la opción de Titulación Curricular en la carrera de:
“Ingeniería en Sistemas Computacionales”

PRESENTAN:

MONTEÓN VALDÉS RAÚL KEVIN
TOVAR BARBA MAYRA PATRICIA

DIRECTORES:



BOTELLO CASTILLO ALEJANDRO

VÉLEZ SALDAÑA ULISES

México, D.F. a 05 de Noviembre de 2018

Índice general

1. Capítulo 1	5
1.1. Introducción	5
1.2. Planteamiento del problema	5
1.2.1. Problemática	5
1.3. Propuesta de solución	6
1.4. Estado del arte	7
1.5. Justificación	8
1.6. Objetivos	9
1.6.1. General	9
1.6.2. Específicos	9
1.7. Arquitectura de la red distribuida	9
1.8. Alcances y Limitaciones	10
1.8.1. Limitaciones	10
1.8.2. Alcances	11
1.8.3. Prototipo 1	11
1.8.4. Prototipo 2	11
2. Marco teórico	13
2.1. Big Data	13
2.1.1. Las 3 Vs del Big Data	13
2.1.2. Casos de uso del Big Data	14
2.2. Minería de datos	14
2.3. Árboles de decisión	14
2.4. ID3 (Iterative Dichotomiser 3)	15
2.4.1. Entropía	16
2.4.2. Ejemplo de ejecución del algoritmo ID3	17
2.5. C4.5	21
2.5.1. Ejemplo de ejecución del algoritmo C4.5	22
2.6. Algoritmo KNN (K-Nearest Neighbors)	27
2.6.1. Ejemplo de ejecución del algoritmo KNN	27
2.7. Algoritmo K-Means	29
2.7.1. Ejemplo de ejecución del algoritmo K-Means	29
2.8. Redes Bayesianas	30
2.9. Ambiente de análisis de datos	31
2.10. Clúster	31
2.11. MapReduce	31
2.12. Hadoop	31

2.12.1. Características principales de Hadoop	31
2.12.2. Arquitectura de Hadoop	33
2.12.3. Arquitectura de un clúster Hadoop	34
2.12.4. Distribuciones Comerciales de Hadoop	34
2.13. Apache Spark	34
2.13.1. Características principales de Spark	35
2.13.2. Ventajas de usar Apache Spark	35
2.13.3. Componentes de Apache Spark	36
2.14. Interfaz Web	36
3. Evaluación y definición de requerimientos de la red distribuida	37
3.1. Descripción del prototipo	37
3.2. Análisis	37
3.2.1. Análisis de la red distribuida	37
3.2.2. Análisis de los datos	38
3.3. Diseño	39
3.3.1. Diseño de la red distribuida	39
3.3.2. Diseño de propuesta de agrupación de acuerdo a los nodos definidos en la red distribuida	40
3.4. Desarrollo	42
3.5. Pruebas	43
4. Integración de requerimientos de la red distribuida	47
4.1. Descripción del prototipo	47
4.2. Análisis	47
4.2.1. Análisis de la adaptación de los datos a la red distribuida	47
4.3. Diseño	47
4.3.1. Diseño de la red distribuida con nodos de datos	47
4.3.2. Diseño de pruebas de obtención de información	48
4.4. Desarrollo	50
4.4.1. Algoritmo de prueba	51
4.5. Pruebas	52
5. Instalador de Luminus	59
5.1. Descripción del prototipo	59
5.2. Análisis	59
5.3. Diseño	60
5.4. Desarrollo	65
5.5. Pruebas	67
6. Conclusiones	71
6.1. Conclusiones	71
7. Trabajo a futuro	73
7.1. Trabajo a futuro	73
7.2. 1.1 hola	87
7.2.1. Resumen	87
7.2.2. Descripción	87
7.2.3. Trayectorias del Caso de Uso	87
8. Anexos	89
8.1. Anexo A: Código JAVA Map-Reduce	89
8.2. Anexo B: Código del instalador	91

CAPÍTULO 1

Capítulo 1

1.1. Introducción

En el siglo XXI, el Big Data ha sido un tema que ha dado mucho de qué hablar, su estudio ha ido en constante crecimiento así como ha servido de gran manera a empresas dedicadas a las tecnologías de la información a generar conocimiento a partir de cantidades masivas de datos que no tienen una estructura muy bien definida. El propio término Big Data (Grandes Datos en español) lo hace entender, la cantidad de datos que se maneja es demasiado grande para ser siquiera almacenada o procesada de manera tradicional [1].

Big Data provee a quien lo usa conocimiento que de otra forma le habría sido imposible obtener, dicho conocimiento permite hacer análisis de los resultados para con ellos obtener ideas que conduzcan a mejores decisiones y movimientos de negocios estratégicos, por lo anterior, Big Data puede llegar a significar una herramienta de alto valor para las empresas que posean la cantidad de datos suficientes para aplicarlo.

1.2. Planteamiento del problema

En la actualidad, el valor de los datos es muy importante para las empresas; cuando se hace un análisis correcto de la información histórica almacenada, se pueden hacer estadísticas de comportamiento e inclusive predicciones de eventos que vendrán en el futuro, las cuales ayudan a la toma de decisiones [2]. Conforme los datos crecen en el tiempo, también la complejidad en el procesamiento de estos datos aumenta. Una de las tendencias de años recientes, llamada Big Data, utiliza un esquema de cómputo distribuido para aprovechar la infraestructura de cómputo disponible en la empresa, y resuelve de manera cooperativa los problemas de análisis de datos [3].

Si esta tendencia es aplicada de manera efectiva, los datos que se generan, pueden significar que las empresas se muevan mucho más rápidamente, sin problemas y de manera más eficiente, ya que son capaces de proporcionar respuestas a preguntas que de otra forma la empresa ni siquiera sabía de su existencia.

Con una cantidad tan grande de información, los datos pueden ser moldeados o probados de cualquier manera que la empresa considere adecuada. Al hacerlo, las organizaciones son capaces de identificar los problemas de una forma más comprensible [4].

1.2.1. Problemática

A pesar de todas las utilidades que ofrece hacer uso de Big Data a las empresas, también se tiene que considerar que su uso implica una gran inversión de recursos económicos, recursos humanos y tiempo.

Las curvas de aprendizaje de las tecnologías con que se hace uso de Big Data suelen ser pronunciadas, por lo que generalmente conlleva un tiempo amplio a quienes quieren aprender a usarlas. Además de que se requiere de una base de conocimiento en cuestión de bases de datos, sistemas operativos, programación, entre otras áreas.

En el ámbito económico, implementar un ambiente de Big Data suele tener un costo económico muy elevado, ya que normalmente se utilizan servidores dedicados. Tanto costo de los equipos mismos, como de su mantenimiento y el acondicionamiento del lugar donde se pondrán a funcionar, significa un costo elevado.

Por otro lado, considerando estas problemáticas, la implementación y puesta en marcha de un ambiente de Big Data también se puede complicar al momento incluir los datos que la empresa desea analizar. Las razones por las que esto comúnmente pasa, se explican a continuación:

- Datos de gran volumen.^[4]
- Los datos cambian rápidamente por lo que su tiempo de validez es muy corto.^[4]
- Debido a que los datos son muy volátiles para que se obtengan resultados valiosos al aplicar Big Data sobre ellos se requiere un poder de procesamiento alto.
- Es necesario discernir entre qué datos son importantes en el análisis y cuáles no lo son.^[4]

1.3. Propuesta de solución

En este trabajo terminal se propone desarrollar un ambiente de Big Data que pueda adaptarse a diferentes casos de estudio o dicho en otras palabras, grupos de datos con diferentes características a analizar, incluyendo un archivo de configuración que permita realizar las adaptaciones correspondientes para cada uno de estos casos de estudio.

El ambiente de Big Data incluirá la aplicación de los siguientes algoritmos de minería de datos:

- Un algoritmo de reglas de asociación
- Un algoritmo de árboles de decisión

Así como la generación de conocimiento y resultados a partir de la aplicación de estos algoritmos.

Haciendo un análisis de los datos correspondientes al caso de estudio, se propondrá el algoritmo de minería de datos que se recomienda para el caso de estudio, sin embargo, se podrá aplicar el algoritmo que el usuario experto desee sin que este análisis lo limite.

Esta solución podrá implementarse en computadoras tradicionales, reduciendo así el costo económico de su implementación en gran medida, ya que se estaría eliminando la necesidad de comprar equipos especializados y ambientarlos. Debido a que el ambiente de Big Data del que hablamos requiere hacer uso, de varias computadoras tradicionales y cada una de ellas requiere llevar a cabo un proceso de instalación y configuración para que este pueda funcionar. Se desarrollará un instalador que permita simplificar esta tarea realizando todo este proceso desde un solo nodo que sera definido como nodo maestro.

La ejecución y puesta en marcha de los algoritmos de minería de datos se hará desde una plataforma web la cual será accedida por el usuario experto desde el navegador web de un ordenador únicamente conociendo la dirección IP asignada al nodo maestro.

1.4. Estado del arte

Actualmente en el mercado existen diferentes alternativas para los usuarios que busquen hacer uso de Big Data. Las alternativas encontradas, podrían dividirse en 2 tipos:

- Software que ya contiene algoritmos de minería de datos implementados.

SOFTWARE	CARACTERÍSTICAS	PRECIO EN EL MERCADO
Knime	<ul style="list-style-type: none"> -Diferentes maneras de presentar los datos como creación de modelos estadísticos y de minería de datos, como árboles de decisión, máquinas de vector soporte, regresiones, etc. -Aplicación de modelos creados con Knime sobre conjuntos nuevos de datos. -Creación de informes -Posible incorporación de nuevos módulos en lenguaje R o Python por el usuario final. <p>Tamaño de instalación: 1.98GB para la versión gratuita [5]</p>	Gratuita (Menos funcionalidad)/ De costo Desde \$44,200 pesos al año por un usuario Hasta \$466,410 pesos al año por 5 usuarios
SPSS Modeler IBM	<ul style="list-style-type: none"> -Posible incorporación de nuevos módulos en lenguaje R Python, Hadoop, Spark u otras tecnologías de código abierto por el usuario final. -Diferentes maneras de presentar los datos mediante la creación de modelos. -Acceso y Exportación de datos sin límite de tamaño. -Más de 30 algoritmos de minería de datos implementados -Ejecuta flujos de datos directamente hacia Spark o Hadoop -Soporte técnico <p>Tamaño de instalación: 1.4GB para la versión de prueba [6]</p>	Gratuita (Prueba 30 días)/De costo Es necesario contactar directamente con IBM para obtener el precio de la herramienta personalizado para cada empresa
Rapid Miner	<ul style="list-style-type: none"> -Cuenta con gráficos para visualización de información -Comparte reutiliza e implementa modelos predictivos -Ejecuta flujos de datos directamente hacia Hadoop -Presentación de datos desde diferentes representaciones gráficas en tiempo real incluso la creación de árboles de decisión [7] 	Desde \$2500.00 pesos al año hasta \$10000.00 pesos al año

- Software de administración de clústers.

SOFTWARE	CARACTERÍSTICAS	PRECIO EN EL MERCADO
Databricks	-Optimizado para lenguaje R. -Muy simple de operar. -Se enfoca en la usabilidad y en procesos intuitivos de navegación. [8]	-Basic - \$0.07 USD por hora de uso de unidad de procesamiento. -Analytics - \$0.2 USD por hora de uso de unidad de procesamiento.[11]
Cloudera Manager	-Plataforma de administración de clústers. -Tiene su propia implementación de búsqueda y ambiente Hadoop con integración y API estándar (CDH). -Cuenta con una interfaz gráfica vía web que opera el stack completo de tecnologías de Big Data. [9]	-Gratuita (Prueba de 60 días) -Essentials \$0.06 USD por hora de uso de unidad de procesamiento -Enterprise Data Hub \$0.87 USD por hora de uso de unidad de procesamiento. [12]
HortonWorks	-Completamente open-source. -Cuenta con un set completo de herramientas necesarias para un ambiente Hadoop. -Se puede lanzar en la nube o en una máquina virtual. -Gran extensibilidad debido a su naturaleza de código abierto. -Utiliza Ambari como interfaz de usuario. [10]	Gratis [13]

Podría decirse que *Luminus* es un híbrido entre las dos clasificaciones anteriormente citadas. Las principales características de *Luminus* son las siguientes:

- Trabaja con datos que se encuentra en distintos equipos de cómputo (Red distribuida).
- El usuario puede seleccionar el algoritmo de minería de datos a utilizar para su funcionamiento dentro de un listado de algoritmos disponibles.
- Es capaz de detectar cuando alguno de los nodos de datos es modificado antes de resolver un determinado problema.
- Los datos son extraídos directamente de los repositorios en tiempo real.
- Se trabaja con archivos JSON de los datos de entrada.

1.5. Justificación

El principal motivo de desarrollo de la presente propuesta es el de facilitar a las empresas u organizaciones la implementación de un ambiente de Big Data, dado que se trata de un proceso complicado, tardado y muy costoso. Cada uno de los factores mencionados anteriormente puede ser justificado de la siguiente manera: la complejidad de la implementación de Big Data se puede reducir mediante la preparación previa del software necesario para la infraestructura que tenga actualmente la empresa; de igual forma se pueden reducir los tiempos de implementación mediante la automatización de las configuraciones y pre procesamiento de datos; y finalmente los costos estarán de acuerdo a las computadoras disponibles (al menos 4) que puedan actuar como nodos de procesamiento y almacenamiento de datos que serán procesados.

Esta herramienta estaría dedicada a empresas u organizaciones que busquen implementar un ambiente de Big Data de una manera más rápida, sencilla, eficiente y a un menor costo [16]. Para que con su adecuado uso, cualquiera de estas entidades sea capaz de mejorar, ser más competitiva, y tener un mayor control sobre su propia información, ya que el conocimiento adquirido a partir de la aplicación de técnicas de minería de datos sobre grandes volúmenes de información estaría siendo de vital importancia para que dichas organizaciones planeen y ejecuten estrategias que les permitan alcanzar sus objetivos [15].

Se ha hecho especial hincapié en el sector empresarial en esta sección, ya que, en general, podría decirse que pequeñas y medianas empresas son los principales clientes potenciales. Esto debido a que la mayoría cuenta con cierta infraestructura de datos y con bitácoras que contienen información sobre sus ingresos y egresos. Toda esa información será la materia prima que será utilizada por la herramienta para la generación de conocimiento.

Si bien, existen algunas implementaciones comerciales como puede verse en la sección **Estado del arte** que realizan la automatización de la implementación de Big Data, son realmente costosas en recursos de cómputo, y el licenciamiento tiene un costo elevado con relación a los usuarios y/o equipos a usar [17]. Algunas implementaciones están basadas en máquinas virtuales, que, si bien, automatizan todo el trabajo, de igual forma resultan costosas con relación al equipo necesario y dinero invertido [14]. Además, el costo de las herramientas comerciales más robustas puede llegar a los cientos de miles de pesos.

1.6. Objetivos

1.6.1. General

Desarrollar un software que permita realizar análisis de datos para la toma de decisiones, mediante la implementación de un ambiente de cómputo distribuido basado en Big Data, y aplicando técnicas de aprendizaje de máquina (machine learning), como reglas de asociación y árboles de decisión, a grandes volúmenes de datos.

1.6.2. Específicos

- Preparación del ambiente de software (instalación, configuración, validación del software del ambiente distribuido) para la implementación de Big Data
- Programación de las rutinas de análisis de datos; algoritmos de aprendizaje de máquina.
- Pruebas y verificación de funcionamiento
- Generación de resultados (reportes y visualizaciones) del análisis de datos.

1.7. Arquitectura de la red distribuida

El siguiente diagrama muestra la arquitectura prevista del sistema:

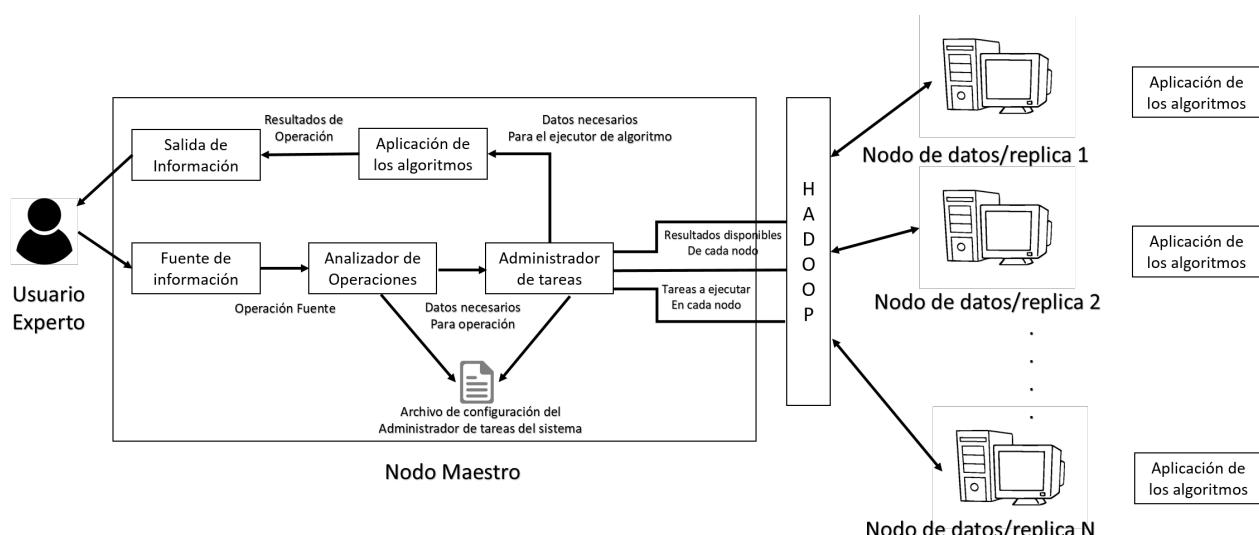


Figura 1.1: Arquitectura de la red distribuida

Con esta arquitectura será posible adaptar el ambiente de Big Data a diversos casos de estudio de las empresas, modificando únicamente archivos que serán destinados para la configuración de dichos casos de estudio.

Esto sin tener que modificar la arquitectura del sistema ni entender su funcionamiento interno. Será posible mantener muy bien separados los bloques de código que realizan cada actividad en el sistema para que en caso de algún fallo sea más fácil detectar en dónde está ocurriendo.

A continuación se explicarán cada uno de los bloques que conforman la arquitectura del sistema.

- **Salida de información:** Es el bloque que se encarga de presentar el resultado final al usuario experto, luego de aplicar los algoritmos de Big Data sobre ellos.
- **Aplicación de los algoritmos:** Es el modulo que permite aplicar los algoritmos de minería de datos sobre los nodos de datos/réplica a los datos. ademas de aplicar la función reduce con los resultados en el nodo maestro. El funcionamiento de la función reduce se explica en la sección del documento [4.3.2](#) dentro del capítulo [Integración de requerimientos de la red distribuida](#).
- **Fuente de información:** Es el modulo que recibe como dato de entrada el algoritmo de minería de datos a ejecutar del usuario conocida como operación fuente.
- **Analizador de operaciones:** Es el modulo encargado de revisar el caso de estudio en cuestión del usuario final para visualizar si es que se tienen que realizar ajustes en el **Archivo de configuración del administrador de tareas** además de revisar que todo se encuentre funcionando de manera adecuada y que sea posible ejecutar el algoritmo de minería de datos en ese momento. Por otro lado también será el encargado de determinar la recomendación del algoritmo de minería de datos mas apropiado para cada caso de estudio.
- **Administrador de tareas:** es el encargado de administrar las tareas que llevará a cabo cada nodo de datos/réplica así como también de recibir los resultados de las tareas que estos ejecuten de manera local en el nodo maestro.
- **Archivo de configuración del administrador de tareas :** Este archivo servirá para llevar a cabo el ajuste de los algoritmos de minería de datos a cada caso de estudio que quiera aplicar el usuario experto. sera necesario realizar algunos ajustes en el mismo para que los algoritmos puedan ser ejecutados de manera satisfactoria.

La arquitectura que se presenta por lo tanto, es una arquitectura del tipo maestro-esclavo, la cual es utilizada para el diseño de un cluster de Hadoop como se detalla en la sección [Hadoop](#) del marco teórico.

El cluster de Hadoop es el que se utiliza para ambientes distribuidos que hacen uso de Big Data dentro de su funcionamiento al igual que Luminus.

1.8. Alcances y Limitaciones

1.8.1. Limitaciones

A continuación se definirán los alcances y limitaciones que tiene este trabajo terminal.

Primeramente las limitaciones que se han detectado hasta ahora en cuanto a equipo de computo para poder ejecutar el ambiente de Big Data por parte del usuario experto:

- Se necesitan al menos 2 computadoras para poder tener un cluster distribuido, por lo que, a pesar de que las tecnologías pueden ejecutar en un solo nodo de trabajo, se perderían las ventajas que ofrece el computo distribuido.
- Es necesario que los equipo que se utilicen cuenten con el triple del espacio de almacenamiento del archivo completo correspondiente al caso de estudio cada uno en disco duro, para que además de permitir el almacenamiento de los datos, también sea posible tener espacio para trabajar con ellos.
- Equipos con 1GB de RAM o menos no tienen un funcionamiento apropiado con Apache Hadoop y su rendimiento es inconsistente por lo que, no se recomienda trabajar en equipos de con estas características.

Por otro lado existen otro tipo de limitaciones que se presentan durante el desarrollo de este trabajo terminal:

- Solo se tendrán 2 algoritmos de minería de datos para ejecutar dentro del ambiente de Big Data, para un estudio mas especializado seria conveniente incluir mas algoritmos.
- Las pruebas y desarrollo sobre este trabajo terminal en equipos de computo tradicionales serán ejecutadas únicamente con 3 nodos de datos/replica, por lo que, para un mejor desempeño sería conveniente agregar mas nodos de datos/replica al cluster.
- El archivo de datos proveniente del caso de estudio es de 21 GB, una empresa podría manejar archivos de mayor longitud, por lo que las pruebas realizadas podrían no ser significativos para la cantidad de datos que puedan manejar determinadas empresas.
- Las tablas de datos que conformen el caso de estudio que se suban a el repositorio , tendrán que subirse por separado y el manejo de los datos que se podrá ejecutar descartará las relaciones que existan entre las tablas y procesará cada tabla por separado.
- Es posible que no se puedan realizar pruebas o bien suficientes pruebas en equipos de computo especializados para realizar una comparativa del comportamiento que tiene Hadoop, bajo diferentes características computacionales.
- El instalador de Luminus tiene muchos casos de errores o casos especiales que no se considerarán y se dejarán a solución del usuario, cuando estos ocurran, ya que por cuestiones de tiempo es imposible englobarlos todos.

Las limitaciones mencionadas no son todas las que existen, solo se mencionan las mas importantes de las que se han encontrado y se considera que tienen relevancia alta para este proyecto.

1.8.2. Alcances

Los alcances que se tienen para los prototipos 1 y 2 que son los prototipos que se desarrollaron en Trabajo Terminal 1 se enlistan a continuación, mientras que los alcances que se tienen para el resto de los prototipos se encuentran definidos en la sección [Trabajo a futuro](#)

1.8.3. Prototipo 1

El prototipo 1 tiene como productos esperados:

- Identificar y reconocer las características de los datos que conforman el caso de estudio así como analizar y definir las posibles agrupaciones que pueden ser generadas a partir de estos datos. Los resultados de este análisis a detalle se encuentran en la sección [Análisis de los datos](#) del capítulo 3.
- Establecer el numero de nodos y características que tendrán los nodos que se agregarán a el cluster para realizar las pruebas del caso de estudio de manera distribuida.
Buscando que con el numero de nodos que se defina se pueda soportar dicho caso de estudio y que además no escapen de las características de computación con las que se cuenta actualmente.
Los resultados obtenidos se pueden consultar en las secciones [Análisis de la red distribuida](#) y [Diseño de la red distribuida](#) del capítulo 3.
- Poner en funcionamiento el cluster con las características que fueron definidas para este en el punto anterior.
El desarrollo que implico dicha tarea se presenta a detalle en la sección [Desarrollo](#) mientras que las pruebas que se hicieron para demostrar que este desarrolle es funcional se pueden consultar en la sección [Pruebas](#) ambas secciones se encuentran en el capítulo 3.

1.8.4. Prototipo 2

El prototipo 2 tiene como productos esperados:

- Cargar los datos que conforman el caso de estudio de prueba a el cluster en funcionamiento que se explica en la sección [1.8.3](#) en su tercer punto. Con lo cual los datos quedarán almacenados de forma distribuida en el cluster.
Para conocer mas detalles de la carga de los datos que se realizo se puede consultar la sección [Desarrollo](#) en el capítulo 4.

- Una vez que los datos son cargados satisfactoriamente a el cluster será necesario comprobar que estos pueden funcionar de manera distribuida y son accesibles, para ello se aplica un algoritmo sencillo sobre los datos. Para conocer mas detalles acerca de esta prueba se puede consultar a partir de la sección **Algoritmo de prueba** en el capitulo 4.

CAPÍTULO 2

Marco teórico

En esta sección se presentan los conceptos que fundamentan este trabajo. Con el objetivo de dar al lector el conocimiento necesario para comprender los capítulos que integran la documentación de *Luminus*.

En la actualidad, la necesidad de almacenamiento de datos crece a niveles exponenciales. Muchas empresas están optando por implementar algún sistema informático para llevar el almacenamiento y gestión de sus datos. Empresas grandes como Google, Facebook, Youtube y demás empresas que diariamente procesan y almacenan cantidades gigantescas de información, requieren usar técnicas un tanto específicas para poder realizar este procesamiento. Ya que al ser tan grande la cantidad de datos, técnicas convencionales serían de poca o nula utilidad para llevar a cabo esa tarea. Una forma de lograrlo se encuentra en un conjunto de técnicas conocidas como *Big Data*.

2.1. Big Data

Al ser el Big Data un concepto tan importante para este trabajo, se procederá a definirlo, ya que como se menciona en el **Capítulo 1**, *Luminus* realizará procesamiento de grandes cantidades de información.

El término Big Data se refiere a cantidades enormes de información, por ejemplo, la cantidad de información que se produce todos los días con el uso de una red social como Facebook, o la cantidad de datos que producen computadoras y dispositivos electrónicos que se auto monitorean mediante sensores. Esos volúmenes masivos de datos pueden ser utilizados para extraer conocimiento de ellos, y posteriormente atacar problemas que no sería posible resolver sin el Big Data.[20]

2.1.1. Las 3 Vs del Big Data

Al ser el Big Data un concepto emergente y relativamente nuevo, se han tenido ciertas dificultades para definirlo de manera uniforme. Debido a las dimensiones de todo lo que conlleva entender Big Data, resultó conveniente entre los estudiosos del tema definir y acentuar las magnitudes que lo definen. Estas magnitudes se conocen como las 3 Vs del Big Data.

1. **Volumen:** Con Big Data, se tendrán que procesar enormes volúmenes de información. Este punto es importante ya que la necesidad de almacenamiento de datos en el mundo moderno, crece de forma exponencial.[20]
2. **Velocidad:** Usando Big Data, se abre la posibilidad de acceso y flujo de datos a velocidades que no se podrían conseguir de manera convencional.[20]
3. **Variedad:** Procesamiento de datos de naturaleza heterogénea, es decir, múltiples tipos de datos.[20]

2.1.2. Casos de uso del Big Data

- **Desarrollo de productos:** Compañías como Netflix y Procter & Gamble utilizan el Big Data para anticiparse a la demanda de los consumidores de sus productos. Utilizan modelos predictivos para sus nuevos productos clasificando atributos clave de sus anteriores productos modelando las relaciones entre esos atributos.[21]
- **Mantenimiento predictivo:** Se pueden predecir fallas mecánicas en maquinaria que de otra forma quedarían fácilmente ignoradas. Mejorando así ampliamente la calidad y el costo del mantenimiento de dichos equipos.[21]
- **Experiencia de usuario:** El uso de Big Data permite recopilar toda la información sobre el usuario y utilizarla a favor de su experiencia en un sitio o aplicación. Por ejemplo, sus búsquedas frecuentes, los sitios que visita, etc. Para de esta manera empezar a hacerle ofertas o anuncios personalizados, según sus intereses particulares.[21]
- **Machine Learning:** Actualmente el machine learning es un área de mucho auge, ya que permite entrenar a las computadoras mediante conjuntos de datos de entrenamiento en lugar de programarlas. El Big Data facilita esa tarea.[21]

2.2. Minería de datos

La minería de datos es el proceso de generar conocimiento a partir de un conjunto de información de gran tamaño. Para generar ese conocimiento se utilizan diferentes técnicas de análisis que detectan patrones y tendencias en la información que se está procesando. Si se intentara utilizar algún método de análisis tradicional, sería muy complicado o incluso imposible a veces encontrar patrones y tendencias útiles, ya que es muy probable que dentro de los datos existan relaciones muy complejas o simplemente la cantidad de datos sea demasiado grande.[22]

La minería de datos puede utilizarse en escenarios como los que se enuncian a continuación:

- **Pronóstico:** Predicción de datos y eventos que vendrán a futuro a partir del comportamiento de conjuntos de datos que se tienen en el presente. Por ejemplo, predicción de ventas y tendencias de compra.[23]
- **Riesgo y probabilidad:** Es un escenario muy común dentro de los negocios de Business Intelligence. Por ejemplo, se llega a utilizar para encontrar puntos de equilibrio probable para escenarios de riesgo. Elección de los mejores clientes para la distribución de correo directo, determinación del punto de equilibrio probable para los escenarios de riesgo, y asignación de probabilidades a diagnósticos y otros resultados.[23]
- **Recomendaciones:** Muy utilizado en sistemas como MercadoLibre o Amazon, en módulos que toman la información de búsqueda de cada usuario, la procesan y le arrojan recomendaciones de productos o servicios.[23]
- **Búsqueda de secuencias:** Al igual que el escenario de **recomendaciones**, se utiliza mucho en sistemas de venta de artículos por internet. Se analiza el orden de los artículos que se meten a un carrito de compra para poder hacer predicciones útiles y generar conocimiento.[23]
- **Agrupación:** Clasificación de los elementos de un conjunto de información para el posterior análisis de sus afinidades.[23]

2.3. Árboles de decisión

Una de las técnicas de análisis que *Luminus* utilizará para generar conocimiento es el uso de **árboles de decisión**. Un árbol de decisión es un modelo de predicción que apoya al proceso de toma de decisiones. Esta herramienta tiene un campo de aplicación extremadamente amplio, pudiendo ir desde el área de finanzas hasta el área del aprendizaje de máquina. A partir de un conjunto de datos de entrada, se construyen los caminos, dentro del árbol, que llevarán a cada una de las decisiones posibles.[24]

Los árboles de decisión están formados por los siguientes elementos:

1. **Nodos:** Un nodo es un punto del proceso en el que de acuerdo a ciertas condiciones o decisiones se redefine el rumbo del camino. Existen dos tipos de nodo[25]:
 - 1.1. **Nodo de decisión:** Es un nodo en el cual se toma una decisión consciente de acuerdo a las necesidades del problema en cuestión. Estos nodos se representan con un cuadrado.
 - 1.2. **Nodo de incertidumbre:** Es un nodo en el que actúan las probabilidades y la heurística para definir el rumbo que tomará el camino que se hará dentro del árbol. Estos nodos se representan con un círculo
2. **Ramas:** Una rama es una de las respuestas o acciones que se toman a partir de la pregunta o escenario que se presentó en un nodo del cual salió esa rama. Una rama es el camino a otro nodo o escenario resultado de la decisión o evento que definió la rama. Este elemento se representa con una línea[25].
3. **Hojas:** Son escenarios finales, ya clasificados. No tienen ramificaciones, y son el resultado final de seguir un camino de decisiones, acciones y probabilidades. Este elemento se representa con un triángulo[25].

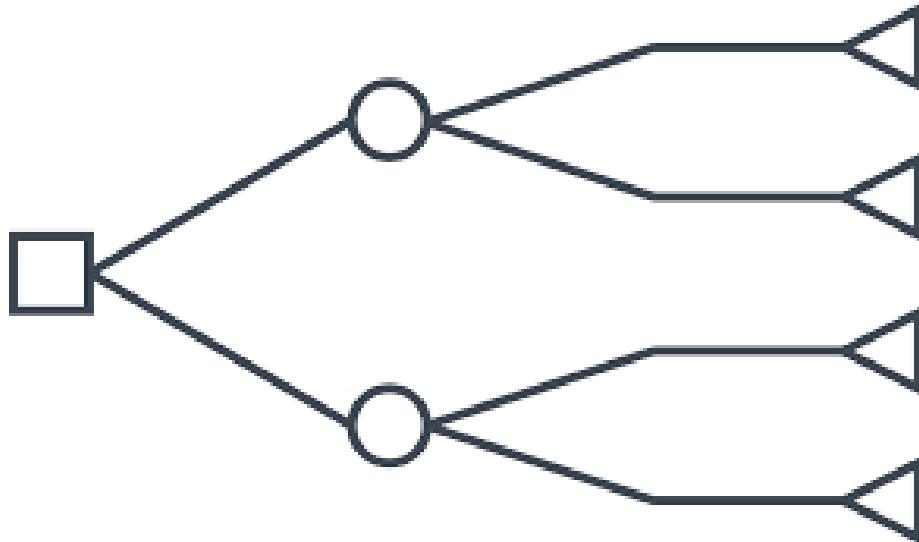


Figura 2.1: Representación general de un árbol de decisión

A continuación se presentan los algoritmos que usan árboles de decisión que tendrá implementados *Luminus* para realizar el análisis de datos.

2.4. ID3 (Iterative Dichotomiser 3)

El algoritmo ID3 es uno de los algoritmos que utilizan árboles de decisión más populares. ID3 genera un árbol a partir de un conjunto de datos llamado **tabla de inducción**. Es útil para hacer toma de decisiones en escenarios binarios, es decir, que tienen 2 posibilidades finales.[26]

Es importante mencionar este algoritmo, no precisamente porque vaya a ser implementado dentro de *Luminus*, sino porque es el precursor de otro algoritmo conocido como **C4.5**. Que es un algoritmo que apoya a la toma de decisiones, más avanzado y completo. Sin embargo, por el momento aún no está bien definido cuál de los dos algoritmos estará implementado en *Luminus*.

El resultado de la ejecución de este algoritmo puede expresarse como un conjunto de reglas **si-entonces**.

2.4.1. Entropía

La entropía es la medida de la aleatoriedad. En otras palabras, la medida e la impredictibilidad. La entropía es más alta cuando todos los eventos posibles en un escenario son igualmente probables, por ejemplo, al tirar una moneda al aire, se tiene 50 % de probabilidad de que caiga cara y 50 % de probabilidad de que caiga cruz, por lo que la entropía es de 1. Este parámetro comienza a decrecer cuando hay una probabilidad o probabilidades que parezcan aplastantes sobre las otras. Las fórmulas para calcular la entropía y la ganancia son las siguientes:

$$\text{Entropia}(S) = \sum_{i=1}^n -p_i \log_2 p_i$$

Figura 2.2: Fórmula para calcular la entropía.

$$\text{Gan Inf}(S, A) = \text{Entropia}(S) - \sum_{v \in V(A)} \frac{|Sv|}{|S|} \text{Entropia}(Sv)$$

Figura 2.3: Fórmula para calcular la ganancia.

2.4.2. Ejemplo de ejecución del algoritmo ID3

Para ilustrar el funcionamiento del algoritmo ID3, utilizaremos la siguiente tabla de inducción que contiene información sobre factores que influyen al tomar la decisión de si un partido de tenis debería o no jugarse:

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
1	Soleado	Caluroso	Alta	Ligero	No
2	Soleado	Caluroso	Alta	Fuerte	No
3	Nublado	Caluroso	Alta	Ligero	Sí
4	Lluvioso	Templado	Alta	Ligero	Sí
5	Lluvioso	Fresco	Normal	Ligero	Sí
6	Lluvioso	Fresco	Normal	Fuerte	No
7	Nublado	Fresco	Normal	Fuerte	Sí
8	Soleado	Templado	Alta	Ligero	No
9	Soleado	Fresco	Normal	Ligero	Sí
10	Lluvioso	Templado	Normal	Ligero	Sí
11	Soleado	Templado	Normal	Fuerte	Sí
12	Nublado	Templado	Alta	Fuerte	Sí
13	Nublado	Caluroso	Normal	Ligero	Sí
14	Lluvioso	Templado	Alta	Fuerte	No

Tabla 2.1: Tabla de inducción para juegos de tenis ID3.

Necesitaremos de la [2.2 fórmula para calcular la entropía](#) y de la [2.3 fórmula para calcular la ganancia](#) para poder proceder con la ejecución del algoritmo.

- Primeramente se calcula la entropía. La columna de **Decisión** consta de 14 instancias e incluye dos posibles valores: **sí** y **no**. Hay 9 decisiones con la etiqueta **sí** y 5 con la etiqueta **no**. Utilizando la fórmula de la entropía, se encuentra que el resultado es de **0.940**.
- Despues de los 5 factores disponibles, se busca el factor más dominante para tomar la decisión. Utilizando la fórmula de la ganancia tomando como parámetros la entropía de la columna **Decisión** y **Viento**.
- El atributo de viento tiene dos posibles valores: **fuerte** y **ligero**. Y al reflejar estos dos posibles valores en la fórmula, ésta se vería más o menos así: $Ganancia(Decisión, Viento) = Entropía(Decisión) - [p(Decisión, Viento = Ligero)*Entropía(Decisión, Viento = Ligero)]-[p(Decisión, Viento = Fuerte)*Entropía(Decisión, Viento = Fuerte)]$.
- Ahora se calcula $(Decisión, Viento = Ligero)$ y $(Decisión, Viento = Fuerte)$ respectivamente.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
1	Soleado	Caluroso	Alta	Ligero	No
3	Nublado	Caluroso	Alta	Ligero	Sí
4	Lluvioso	Templado	Alta	Ligero	Sí
5	Lluvioso	Fresco	Normal	Ligero	Sí
8	Soleado	Templado	Alta	Ligero	No
9	Soleado	Fresco	Normal	Ligero	Sí
10	Lluvioso	Templado	Normal	Ligero	Sí
13	Nublado	Caluroso	Normal	Ligero	Sí

Tabla 2.2: Tabla de para el factor de decisión *Viento Ligero*.

Dentro de la tabla de viento ligero hay 8 instancias en total, de las cuales 2 tienen la decisión final **no** y 6 tienen la decisión final **sí**. Al introducir estos datos en la [2.2 fórmula de la entropía](#), y obtenemos una entropía de **0.811**.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
2	Soleado	Caluroso	Alta	Fuerte	No
6	Lluvioso	Fresco	Normal	Fuerte	No
7	Nublado	Fresco	Normal	Fuerte	Sí
11	Soleado	Templado	Normal	Fuerte	Sí
12	Nublado	Templado	Alta	Fuerte	Sí
14	Lluvioso	Templado	Alta	Fuerte	No

Tabla 2.3: Tabla de para el factor de decisión *Viento Fuerte*.

En la tabla de viento fuerte encontramos 6 instancias divididas en 2 partes iguales: 3 tienen la decisión final **sí** y 3 tienen la decisión final **no**. Al sustituir estos datos en la [2.2 fórmula de la entropía](#), obtenemos una entropía de 1.

- Ahora que tenemos esos dos valores, podemos volver a la ecuación de la ganancia. El resultado queda expresado como $Ganancia(Decisión, Viento) = 0.940 - [(8/14)*0.811] - [(6/14)*1] = 0.048$.

- En este punto se ha concluido ya el cálculo para el factor de decisión *Viento*. Ahora se necesita hacer lo mismo para todas las demás columnas/factores.

- A modo de resumen:

1. $Ganancia(Decisión, Aspecto) = 0.246$

2. $Ganancia(Decisión, Temperatura) = 0.029$

3. $Ganancia(Decisión, Humedad) = 0.151$

- Como podemos ver, el factor de decisión *Aspecto* es el que produce el valor de ganancia más alto. Por tal motivo, ese factor de decisión será el nodo raíz del árbol de decisión.

- Ahora debemos probar todos los valores posibles que tiene el factor de decisión *Aspecto*:

- Aspecto Nublado

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
3	Nublado	Caluroso	Alta	Ligero	Sí
7	Nublado	Fresco	Normal	Fuerte	Sí
12	Nublado	Templado	Alta	Fuerte	Sí
13	Nublado	Caluroso	Normal	Ligero	Sí

Tabla 2.4: Tabla de días con aspecto *nublado*.

Podemos observar que siempre que el aspecto del día sea **nublado**, la decisión final sería **sí**.

- Aspecto Soleado

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
1	Soleado	Caluroso	Alta	Ligero	No
2	Soleado	Caluroso	Alta	Fuerte	No
8	Soleado	Templado	Alta	Ligero	No
9	Soleado	Fresco	Normal	Ligero	Sí
11	Soleado	Templado	Normal	Fuerte	Sí

Tabla 2.5: Tabla días con aspecto *soleado*.

En esta tabla tenemos 5 instancias para el aspecto **soleado**. Hay 3 de esas instancias que tienen como decisión final **sí** y 2 que tienen **no**. Por lo que los valores de ganancia del factor de decisión **Aspecto Soleado** con respecto a todos los demás factores de decisión quedarán de la siguiente manera:

1. Ganancia(Aspecto = Soleado, Temperatura) = 0.570
2. Ganancia(Aspecto = Soleado, Humedad) = 0.970
3. Ganancia(Aspecto = Soleado, Viento) = 0.019

En este punto, la humedad es el factor de decisión con mayor ganancia cuando el aspecto del día es soleado. Por lo que debemos probar todos los valores posibles para el factor de decisión *humedad*.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
1	Soleado	Caluroso	Alta	Ligero	No
2	Soleado	Caluroso	Alta	Fuerte	No
8	Soleado	Templado	Alta	Ligero	No

Tabla 2.6: Tabla días con aspecto *soleado* y humedad *alta*.

La decisión siempre será **no** cuando la humedad sea **alta**.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
9	Soleado	Fresco	Normal	Ligero	Sí
11	Soleado	Templado	Normal	Fuerte	Sí

Tabla 2.7: Tabla días con aspecto *soleado* y humedad *normal*.

Por otro lado, la decisión siempre será **sí** cuando la humedad es **normal**.

De lo anterior concluimos que deberemos verificar la humedad y decidir si el aspecto del día es **soleado**.

- Aspecto Lluvioso

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
4	Lluvioso	Templado	Alta	Ligero	Sí
5	Lluvioso	Fresco	Normal	Ligero	Sí
6	Lluvioso	Fresco	Normal	Fuerte	No
10	Lluvioso	Templado	Normal	Ligero	Sí
14	Lluvioso	Templado	Alta	Fuerte	No

Tabla 2.8: Tabla de días con aspecto *lluvioso*.

Al evaluar los valores de ganancia de los días con aspecto **lluvioso** con respecto a los demás factores de decisión, se encuentra que el factor que genera una mayor ganancia es el viento. Por lo cual se tienen que checar todos los posibles valores de ese factor de decisión.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
4	Lluvioso	Templado	Alta	Ligero	Sí
5	Lluvioso	Fresco	Normal	Ligero	Sí
10	Lluvioso	Templado	Normal	Ligero	Sí

Tabla 2.9: Tabla de días con aspecto *lluvioso* y con viento *ligero*.

De la tabla de aspecto **lluvioso** y viento **ligero** podemos deducir que siempre que existan estas dos condiciones al mismo tiempo, la decisión final será **sí**.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
6	Lluvioso	Fresco	Normal	Fuerte	No
14	Lluvioso	Templado	Alta	Fuerte	No

Tabla 2.10: Tabla de días con aspecto *lluvioso*.

De la tabla de aspecto **lluvioso** y viento **fuerte** podemos deducir que siempre que existan estas dos condiciones al mismo tiempo, la decisión final será **no**.

- Finalmente la construcción de este árbol de decisión es la siguiente:

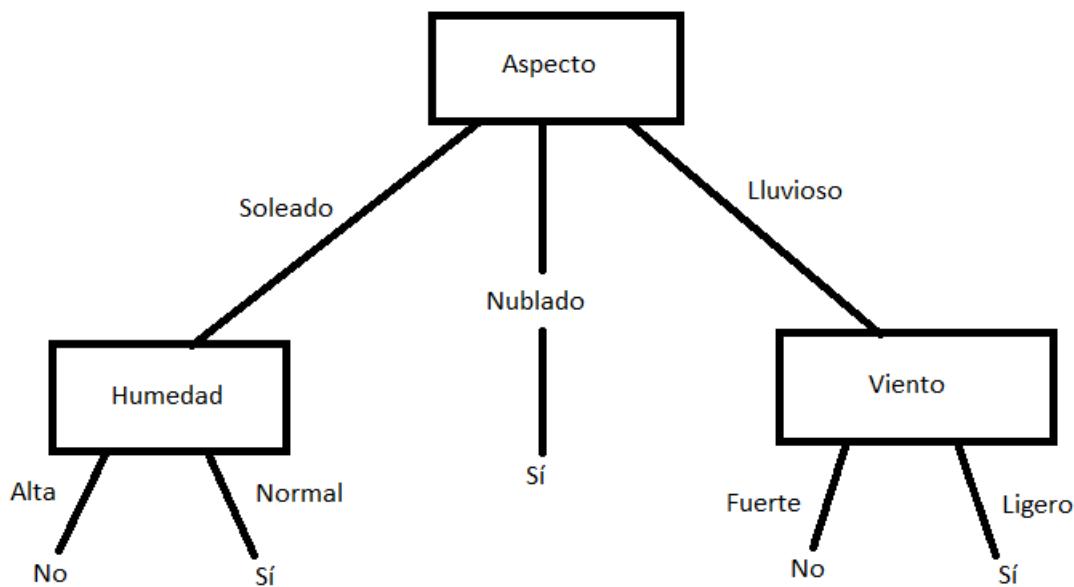


Figura 2.4: Árbol de decisión resultante.

2.5. C4.5

El algoritmo C4.5 es la evolución del algoritmo ID3. Éste genera un árbol de decisión a partir de un conjunto de datos de entrada de manera recursiva, al igual que su precursor^[27]. Sin embargo, aunque ID3 y C4.5 son algoritmos muy semejantes, existen ciertas diferencias:

- C4.5 permite trabajar con valores continuos, mientras que ID3 solamente trabaja con valores discretos.
- Los árboles de C4.5 son más compactos, y esto se debe a que cada nodo hoja engloba un conjunto de clases y no una sola clase particular.
- C4.5 es más eficiente computacionalmente hablando.
- C4.5 utiliza un nuevo parámetro llamado **Gain Ratio**, en lugar de la ganancia simple. Y este a su vez requiere de la ganancia y de otro parámetro nuevo llamado **SplitInfo** cuyas fórmulas se enuncian a continuación^[27]:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

Figura 2.5: Fórmula para obtener el Gain Ratio

$$SplitInfo_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Figura 2.6: Fórmula para obtener SplitInfo

2.5.1. Ejemplo de ejecución del algoritmo C4.5

Para mostrar la forma en la que se ejecuta el algoritmo C4.5 y los pasos que se deben llevar a cabo, se utilizará la siguiente tabla de inducción:

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
1	Soleado	85	85	Ligero	No
2	Soleado	80	90	Fuerte	No
3	Nublado	83	78	Ligero	Sí
4	Lluvioso	70	96	Ligero	Sí
5	Lluvioso	68	80	Ligero	Sí
6	Lluvioso	65	70	Fuerte	No
7	Nublado	64	65	Fuerte	Sí
8	Soleado	72	95	Ligero	No
9	Soleado	69	70	Ligero	Sí
10	Lluvioso	75	80	Ligero	Sí
11	Soleado	75	70	Fuerte	Sí
12	Nublado	72	90	Fuerte	Sí
13	Nublado	81	75	Ligero	Sí
14	Lluvioso	80	80	Fuerte	No

Tabla 2.11: Tabla de inducción para juegos de tenis C4.5.

- Al igual que con el ejemplo [Ejemplo de ejecución del algoritmo ID3](#), lo primero que se hace es calcular la entropía general. Hay 15 instancias de las cuales 9 tienen una decisión final de **sí** y 5 tienen **no**. Al sustituir los valores correspondientes en la [2.2 fórmula para calcular la entropía](#) el resultado que obtenemos es **0.940**.

- En C4.5 se utilizan **Gain Ratios** (radios de ganancia), mientras que en ID3 se utilizan ganancias.
- Empezaremos por analizar el atributo de **Viento**.

Se tienen 8 instancias de *viento ligero*, dos de ellas concluyen en un **no**, y las otras 6 concluyen en **sí** por lo que:

- Entropía(Decisión, Viento = Ligero) = 0.811

2. Entropía(Decisión, Viento = Fuerte) = 1

3. Ganancia(Decisión, Viento) = 0.049

Existen 6 instancias de viento fuerte por lo que:

1. SplitInfo(Decisión, Viento) = 0.985

2. GainRatio(Decisión, Viento) = Gain(Decisión, Viento) / SplitInfo(Decision, Viento) = 0.049

• Continuamos analizando ahora el atributo **Aspecto**.

Se tienen 5 instancias para el aspecto *soleado*, de las cuales 3 concluyen en **no** y las otras 2 concluyen en **sí**. Calculando sus valores de entropías, ganancia, SplitInfo y GainRatio:

1. Entropía(Decisión, Aspecto = Soleado) = 0.970

2. Entropía(Decisión, Aspecto = Nublado) = 0

3. Entropía(Decisión, Aspecto = Lluvioso) = 0.970

4. Ganancia(Decisión, Aspecto) = 0.246

Hay 5 instancias para *soleado*, 4 instancias para *nublado* y 5 para *lluvioso*, por lo que:

5. SplitInfo(Decisión, Aspecto) = 1.577

6. GainRatio(Decisión, Aspecto) = 0.155

• Procedemos a analizar el atributo de humedad. Cuyo caso es diferente al de los demás atributos, ya que este es un atributo continuo. Necesitamos convertir valoress continuos a valores nominales (como todos los demás atributos). C4.5 propone hacer una división binaria a partir de algún valor que podamos tomar como umbral. El umbral debe ser el valor que mayor ganancia ofrezca para ese atributo. Para esto, primero se deben ordenar las instancias de humedad de menor a mayor.

Día	Humedad	Decisión
7	65	Sí
6	70	No
9	70	Sí
11	70	Sí
13	75	Sí
3	78	Sí
5	80	Sí
10	80	Sí
14	80	No
1	85	No
2	90	No
12	90	Sí
8	95	No
4	96	Sí

Tabla 2.12: Tabla de I atributo *humedad* ordenada de menor a mayor.

Ahora debemos recorrer todos los valores de humedad y separar el conjunto de datos en dos partes. Se calcularán la **Ganancia** y el **GainRatio** y el valor que maximice la ganancia será el umbral.

1. Se propone 65 como umbral.
 - 1.1. Entropía(Decisión, Humedad ≤ 65) = 0
 - 1.2. Entropía(Decisión, Humedad > 65) = 0.961
 - 1.3. Ganancia(Decisión, Humedad $\neq 65$) = 0.048
 - 1.4. SplitInfo(Decisión, Humedad $\neq 65$) = 0.371
 - 1.5. GainRatio(Decisión, Humedad $\neq 65$) = 0.126
2. Se propone 70 como umbral.
 - 2.1. Entropía(Decisión, Humedad ≤ 70) = 0.811
 - 2.2. Entropía(Decisión, Humedad > 70) = 0.970
 - 2.3. Ganancia(Decisión, Humedad $\neq 70$) = 0.014
 - 2.4. SplitInfo(Decisión, Humedad $\neq 70$) = 0.863
 - 2.5. GainRatio(Decisión, Humedad $\neq 70$) = 0.016
3. Se propone 75 como umbral.
 - 3.1. Entropía(Decisión, Humedad ≤ 75) = 0.721
 - 3.2. Entropía(Decisión, Humedad > 75) = 0.991
 - 3.3. Ganancia(Decisión, Humedad $\neq 75$) = 0.045
 - 3.4. SplitInfo(Decisión, Humedad $\neq 75$) = 0.940
 - 3.5. GainRatio(Decisión, Humedad $\neq 75$) = 0.047
4. Se continúa haciendo lo mismo para cada valor nuevo de humedad que se vaya encontrando en la tabla. Resumiendo:
5. Ganancia(Decisión, Humedad $\neq 78$) = 0.090

6. Ganancia(Decisión, Humedad $<>80$) = 0.107

7. Ganancia(Decisión, Humedad $<>85$) = 0.027

8. Ganancia(Decisión, Humedad $<>90$) = 0.016

- Como podemos ver, el valor que maximiza la ganancia es el de 80. Lo que significa que ahora se debe comparar los otros valores nominales con el valor 80 del atributo *humedad* para crear una rama en nuestro árbol. De esta forma podemos resumir los resultados en la siguiente tabla:

Atributo	Ganancia	GainRatio
Viento	0.049	0.049
Aspecto	0.246	0.155
Humedad $<>80$	0.101	0.107

Tabla 2.13: Comparación de las ganancias entre atributos.

- Al igual que en el ejemplo del algoritmo ID3, el atributo **aspecto** es el nodo raíz del árbol de decisión, por lo que ahora se deben analizar todos los posibles valores que puede adquirir dicho atributo.

- Aspecto Soleado.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
1	Soleado	85	85	Ligero	No
2	Soleado	80	90	Fuerte	No
8	Soleado	72	95	Ligero	No
9	Soleado	69	70	Ligero	Sí
11	Soleado	75	70	Fuerte	Sí

Tabla 2.14: Tabla de inducción para el atributo *Aspecto* con el valor *Soleado*.

Ya hemos dividido la humedad a partir de su punto de umbral que es 80. Podemos observar en la siguiente tabla que la decisión final será **no**, si la humedad es mayor a 80 y el aspecto del día es soleado. La decisión final será **sí**, si la humedad es menor o igual a 80 para un día soleado.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
3	Nublado	83	78	Ligero	Sí
7	Nublado	64	65	Fuerte	Sí
12	Nublado	72	90	Fuerte	Sí
13	Nublado	81	75	Ligero	Sí

Tabla 2.15: Tabla de inducción para el atributo *Aspecto* con el valor *Nublado*.

Cuando el aspecto del día es **nublado**, no importa ninguna otra condición; ni temperatura, ni humedad. La decisión final siempre será **sí**.

Día	Aspecto	Temperatura	Humedad	Viento	Decisión
4	Lluvioso	70	96	Ligero	Sí
5	Lluvioso	68	80	Ligero	Sí
6	Lluvioso	65	70	Fuerte	No
10	Lluvioso	75	80	Ligero	Sí
14	Lluvioso	80	80	Fuerte	No

Tabla 2.16: Tabla de inducción para el atributo *Aspecto* con el valor *Lluvioso*.

Cuando el aspecto es **lluvioso**, la decisión será **sí** cuando el viento es ligero, y será **no** cuando sea fuerte.

- De todo lo anterior se concluye un árbol de decisión con la siguiente estructura:

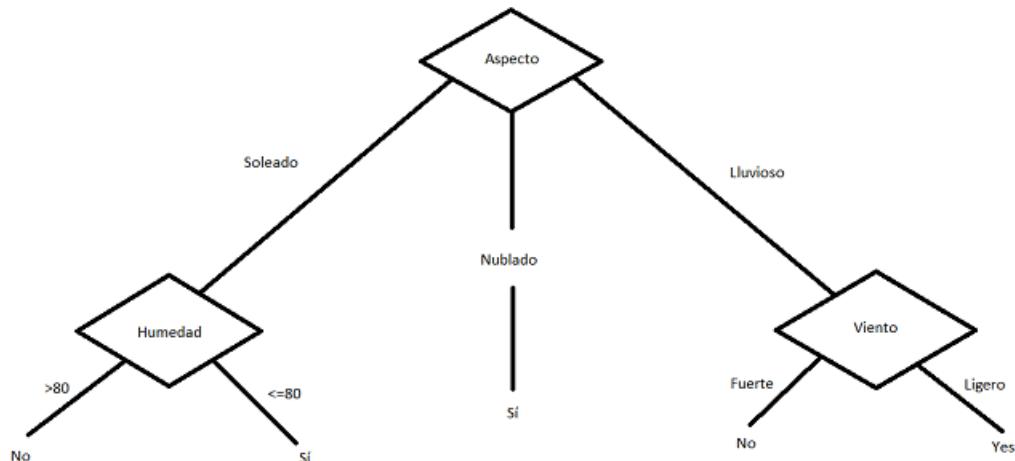


Figura 2.7: Árbol de decisión construido con C4.5.

2.6. Algoritmo KNN (K-Nearest Neighbors)

El algoritmo KNN es un algoritmo de aprendizaje no supervisado en el que se busca clasificar un punto en una categoría con la ayuda de un conjunto de entrenamiento [28]. Las siglas **KNN** en inglés hacen alusión a **K-Nearest Neighbors**, que significa *K* vecinos más cercanos. Este algoritmo calcula distancias euclidianas entre valores que son tomados como puntos. **KNN** estará implementado en *Luminus*, ya que para el caso de estudio que se presentará, será necesario conocer cuáles son las tiendas más cercanas a la ubicación desde donde se esté utilizando el sistema.

Los pasos que sigue el algoritmo KNN se enumeran a continuación:

1. Se calcula la similitud entre puntos basándose en una función de distancia.
2. Se encuentran los *K* vecinos más cercanos.

2.6.1. Ejemplo de ejecución del algoritmo KNN

Con base en la siguiente información que relaciona el peso, la altura y la talla de playera de personas, se hará la ejecución del algoritmo KNN.

Altura (cm)	Peso (kg)	Talla
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Tabla 2.17: Conjunto de entrenamiento para el algoritmo KNN.

Nuevo dato: Altura = 161 cm, Peso = 61 kg

- Se calcula la distancia Euclidiana entre el dato de entrada y cada uno de los datos del conjunto de entrenamiento, utilizando la siguiente fórmula.

$$d_E(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}.$$

Figura 2.8: Fórmula para calcular la distancia Euclíadiana.

Obtenemos lo siguiente:

Altura (cm)	Peso (kg)	Talla	Distancia
158	58	M	4.2
158	59	M	3.6
158	63	M	3.6
160	59	M	2.2
160	60	M	1.4
163	60	M	2.2
163	61	M	2.0
160	64	L	3.2
163	64	L	3.6
165	61	L	4.0
165	62	L	4.1
165	65	L	5.7
168	62	L	7.1
168	63	L	7.3
168	66	L	8.6
170	63	L	9.2
170	64	L	9.5
170	68	L	11.4

Tabla 2.18: Tabla del conjunto de entrenamiento con la columna **Distancia** ya calculada.

- Se toma la distancia más pequeña para determinar al vecino más cercano de entre todas las distancias Euclidianas calculadas previamente. Esta distancia es **1.4**.
- Se define K como el número de vecinos más cercanos que queramos conocer. Para este caso definiremos K igual a 5. Por lo que se tomarán los 5 valores más pequeños.

Altura (cm)	Peso (kg)	Talla	Distancia
160	60	M	1.4
163	61	M	2.0
163	60	M	2.2
160	59	M	2.2
160	64	L	3.2

Tabla 2.19: Instancias de la tabla con menor distancia al nuevo dato.

2.7. Algoritmo K-Means

El algoritmo K-Means es un algoritmo de agrupamiento que utiliza aprendizaje no supervisado, el cual es utilizado cuando se tienen datos no etiquetados, es decir sin categorías o grupos definidos. El objetivo de este algoritmo es encontrar grupos en los datos, con el número de grupos se representa la variable K [29]. Es importante definir este algoritmo y su funcionamiento dentro del marco teórico, ya que es uno de los candidatos a ser implementados dentro de *Luminus*.

El algoritmo arrojará como resultado final lo siguiente:

1. Los centroides de los K grupos.
2. Etiquetas para los datos de entrenamiento. Cada dato es asignado a un solo grupo.

2.7.1. Ejemplo de ejecución del algoritmo K-Means

Para ilustrar el funcionamiento del algoritmo K-means, se utilizará el siguiente conjunto de entrenamiento correspondiente a la puntuación de 7 individuos en 2 pruebas:

Individuo	Prueba 1	Prueba 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Tabla 2.20: Conjunto de entrenamiento para el algoritmo K-Means.

- El conjunto de datos se agrupa en dos grupos distintos. Para esto, se toman los valores de la *Prueba 1* y *Prueba 2* de los individuos que tengan estos valores más lejanos, es decir:

Grupo	Individuo	Vector (Centroide)
Grupo 1	1	(1.0, 1.0)
Grupo 2	4	(5.0, 7.0)

Tabla 2.21: Partición inicial.

- Los individuos restantes ahora son examinados secuencialmente y son ubicados en el grupo al que son más cercanos en términos de distancia Euclidiana. El vector es recalculado cada que un nuevo miembro es agregado.

Iteración	Individuo	Vector (Centroide)
1	1	(1.0, 1.0)
2	1, 2	(1.2, 1.5)
3	1, 2, 3	(1.8, 2.3)
4	1, 2, 3	(1.8, 2.3)
5	1, 2, 3	(1.8, 2.3)
6	1, 2, 3	(1.8, 2.3)

Tabla 2.22: Tabla de individuos agregados secuencialmente al **Grupo 1**.

Iteración	Individuo	Vector (Centroide)
1	4	(5.0, 7.0)
2	4	(5.0, 7.0)
3	4	(5.0, 7.0)
4	4, 5	(4.2, 6.0)
5	4, 5, 6	(4.3, 5.7)
6	4, 5, 6, 7	(4.1, 5.4)

Tabla 2.23: Tabla de individuos agregados secuencialmente al **Grupo 2**.

- Ahora la partición inicial ha cambiado, y los dos grupos tienen las siguientes características:

Grupo	Individuo	Vector (Centroide)
Grupo 1	1, 2, 3	(1.8, 2.3)
Grupo 2	4, 5, 6, 7	(4.1, 5.4)

Tabla 2.24: Partición inicial modificada.

- Sin embargo, no podemos asegurarnos de que esas son las clasificaciones correctas para cada dato. Así que comparamos la distancia de cada individuo a su propio grupo y al grupo opuesto:

Individuo	Distancia al Grupo 1	Distancia al Grupo 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Tabla 2.25: Cálculo de distancia de cada dato a su grupo y al opuesto.

- Solamente el individuo 3 está más cerca al centroide del grupo opuesto (Grupo 2) que de su propio grupo (Grupo 1). Por lo que el individuo 3 se reubica en el Grupo 2. Resultando la siguiente partición:

Grupo	Individuo	Vector (Centroide)
Grupo 1	1, 2	(1.8, 1.5)
Grupo 2	3, 4, 5, 6, 7	(3.9, 5.1)

Tabla 2.26: Partición final.

- Las iteraciones continuarían hasta que ya no haya más reubicaciones.
- Es probable que el algoritmo no encuentre una solución final.

2.8. Redes Bayesianas

Otro concepto que es importante definir es el de las redes Bayesianas, ya que se pretende que *Luminus* pueda hacer uso de esta técnica para generar conocimiento para el usuario.

Las redes Bayesianas son un tipo de modelo probabilístico gráfico que usa inferencias Bayesianas para el cómputo de probabilidades. Las redes Bayesianas pretenden modelar dependencia condicional representando esta dependencia mediante un grafo dirigido. A través de esas relaciones, se puede conducir eficientemente la inferencia de variables aleatorias.[36]

2.9. Ambiente de análisis de datos

Para definir **ambiente de análisis de datos** primero se partirá de la definición de **ambiente de cómputo**.

Un ambiente de cómputo es una colección de computadoras que realizan procesamiento e intercambio de información para resolver diversos tipos de problemas de cómputo.[35]

Partiendo de la definición anteriormente citada, Un ambiente de análisis de datos es una colección de computadoras que tienen como fin principal ejecutar algoritmos de que realicen análisis de conjuntos de datos. *Luminus* es un ambiente de análisis de datos.

2.10. Clúster

Un clúster, en términos de informática, es un conjunto de computadoras interconectadas que pueden ser vistas como un solo sistema. Las computadoras que componen un clúster realizan una tarea en común y son controladas mediante un software específico. Usualmente los clústers son implementados con fines de obtener alto rendimiento en el procesamiento de datos.[34]

Al ser *Luminus* un **Ambiente de análisis de datos** que aplicará técnicas de análisis sobre grandes cantidades de información, se requiere implementarlo en un clúster en el que las computadoras que lo integren trabajen en conjunto para llevar a cabo la ejecución de los algoritmos implementados dentro del sistema y para el almacenamiento distribuido de los datos.

2.11. MapReduce

MapReduce es un paradigma de programación que permite escalabilidad masiva a través de cientos o miles de servidores en un clúster Hadoop. *MapReduce* se refiere a dos tareas distintas y separadas. La primera, *Map*, consiste en realizar mapeos. Ésta convierte un conjunto de datos en otro conjunto de datos diferente en el que los elementos individuales son separados en tuplas. La segunda, *Reduce*, toma los datos que arroja de *Map*, y combina esas tuplas en un conjunto más pequeño de tuplas. MapReduce es el corazón de Hadoop.[30] Los algoritmos implementados dentro de *Luminus* estarán programados con base en este paradigma.

2.12. Hadoop

Una de las capas de la pila de tecnologías que se utilizarán para construir *Luminus* es Hadoop. Hadoop un framework de software que soporta aplicaciones distribuidas bajo una licencia libre. Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos. Hadoop se inspiró en los documentos Google para MapReduce y Google File System (GFS). Hadoop utiliza su propio sistema de archivos HDFS, que divide archivos grandes y los distribuye en diferentes nodos para su procesamiento.[31]

2.12.1. Características principales de Hadoop

- **Procesamiento distribuido:** Hadoop distribuye los datos en los diferentes nodos que formen parte de la arquitectura de clúster que estén haciendo uso de él. Pretende paralelizar tareas de procesamiento de datos[31].
- **Eficiencia:** Mediante la paralelización, se consigue una ganancia considerable en el tiempo de procesamiento de información.[31]
- **Económico:** Propicia un ambiente fácilmente escalable en el que resulta sencillo añadir nodos de manera horizontal conforme se vaya requiriendo.[31]
- **Código abierto:** Es un proyecto de los llamados **open source**.[31]
- **Tolerancia a fallos:** Utiliza replicación de datos haciendo uso de **HDFS (Hadoop Distributed File System)**. Si un nodo falla o cae, hay nodos de respaldo que permiten mantener el ambiente en funcionamiento.[31]

2.12.2. Arquitectura de Hadoop

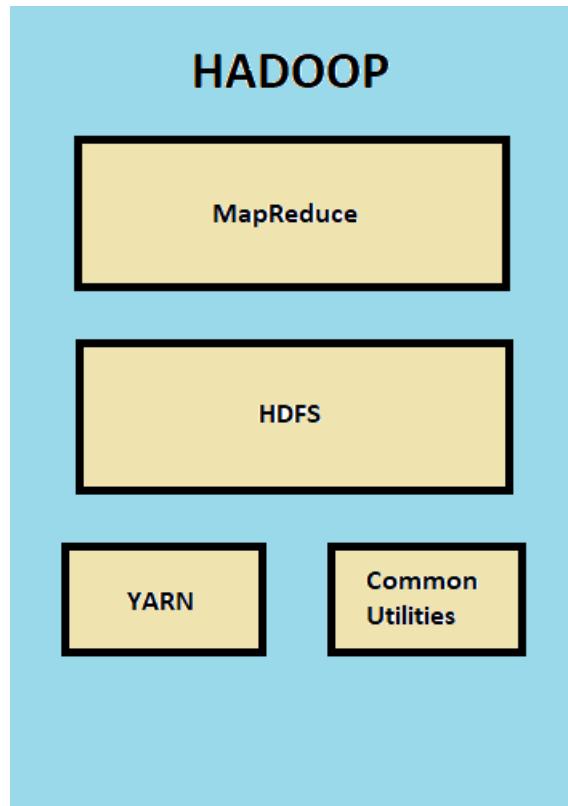


Figura 2.9: Arquitectura básica de Hadoop.

- **MapReduce:** Es el módulo que permite ejecutar cómputo distribuido. Pretende la paralelización de procesos.

- **HDFS (Hadoop Distributed File System:)** Es el sistema de archivos distribuido que utiliza Hadoop. Éste parte los datos en fragmentos y los almacena en los nodos que conforman el clúster donde Hadoop esté ejecutándose.

- **YARN:** Es el gestor de recursos de Hadoop.

- **Common Utilities:** Librerías y código necesario para ejecutar Hadoop.

2.12.3. Arquitectura de un clúster Hadoop

Habitualmente, un clúster Hadoop tiene la estructura **maestro - esclavo**. Lo que significa que hay un nodo **maestro** que estará coordinando la ejecución de tareas y las asignará a los nodos **esclavos**.

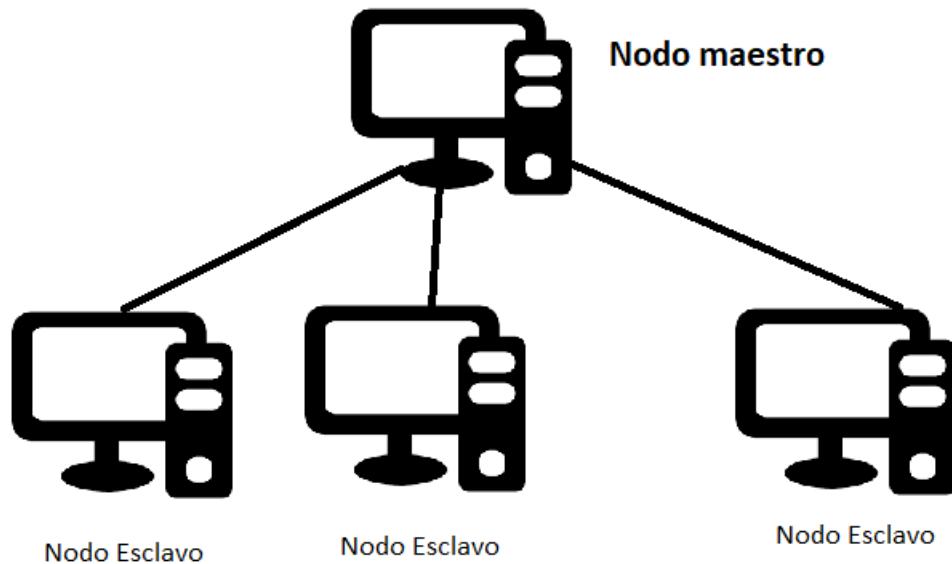


Figura 2.10: Arquitectura de maestros y esclavos.

2.12.4. Distribuciones Comerciales de Hadoop

Durante el proceso de desarrollo de *Luminus* se tenía pensado utilizar alguna distribución comercial de Hadoop, por lo que es importante denotarlas en esta sección, aún cuando al final ninguna de ellas se haya adaptado a las necesidades del proyecto. Existen varias opciones actualmente en el mercado para hacer uso de las capacidades de Hadoop, entre las cuales destacan 3 principalmente[33]:

- **Cloudera:** Fue la primera de todas las distribuciones comerciales de Hadoop. Cuenta con una herramienta llamada Cloudera Manager que permite gestionar el clúster.
- **Hortonworks:** Es la distribución de Hadoop más nueva. Permite instalar el clúster mediante un cliente de virtualización.
- **Microsoft Azure:** Es una distribución desarrollada por Microsoft que permite tener las máquinas del clúster en la nube.

2.13. Apache Spark

Es un motor de análisis de datos unificado para Big Data y Machine Learning. Que se basa en Hadoop Map Reduce. Es una herramienta de código abierto que permite **dividir y ejecutar tareas de manera paralela**. Esta cualidad de Spark de poder paralelizar el trabajo se debe a que éste software en la gran mayoría de los casos es ejecutado en sistemas con arquitectura de clúster.[32] *Luminus* utiliza Spark como motor de análisis de datos, y para la construcción de la red distribuida.

2.13.1. Características principales de Spark

- Integrado con Apache Hadoop.

- Ofrece un desempeño veloz.

- El almacenamiento de datos se administra en memoria. Se reducen mucho los tiempos de ejecución ya que hay muchas menos operaciones de lectura y escritura en disco.

- Puede ejecutar algoritmos escritos en Java, Scala, Python y R.

- Permite procesamiento en tiempo real.

2.13.2. Ventajas de usar Apache Spark

Utilizar Spark para el procesamiento de volúmenes de información de gran tamaño ofrece varios beneficios.[\[32\]](#) Los principales beneficios son los siguientes:

- **Velocidad:** Gran velocidad en el procesamiento de información. Esto se debe a lo ya mencionado anteriormente sobre la gestión de datos desde memoria.

- **Potencia:** Spark nos permite aprovechar el hardware de los equipos que lo estén utilizando.

- **Fácil uso:** A diferencia de Hadoop, que requería de un amplio conocimiento de MapReduce y de Java, Spark permite usar lenguajes de más alto nivel como Python y Scala además de Java.

- **Integración SQL:** Como resultado de contener el módulo Spark SQL, es posible realizar consultas en conjuntos de datos semi-estructurados utilizando lenguaje SQL.

- **Constante mejora del propio sistema:** Al ser un proyecto de código abierto, cada vez hay más personas que contribuyen a la mejora y ampliación de los alcances de Spark.

- **Escalabilidad:** Spark permite que incrementar el tamaño del clúster conforme se vaya necesitando.

2.13.3. Componentes de Apache Spark

Podría decirse que Spark es un conjunto de módulos que nos permiten generar conocimiento usando diferentes técnicas y tecnologías[32].

El diagrama mostrado a continuación ilustra los principales módulos o **componentes** que conforman Apache Spark.

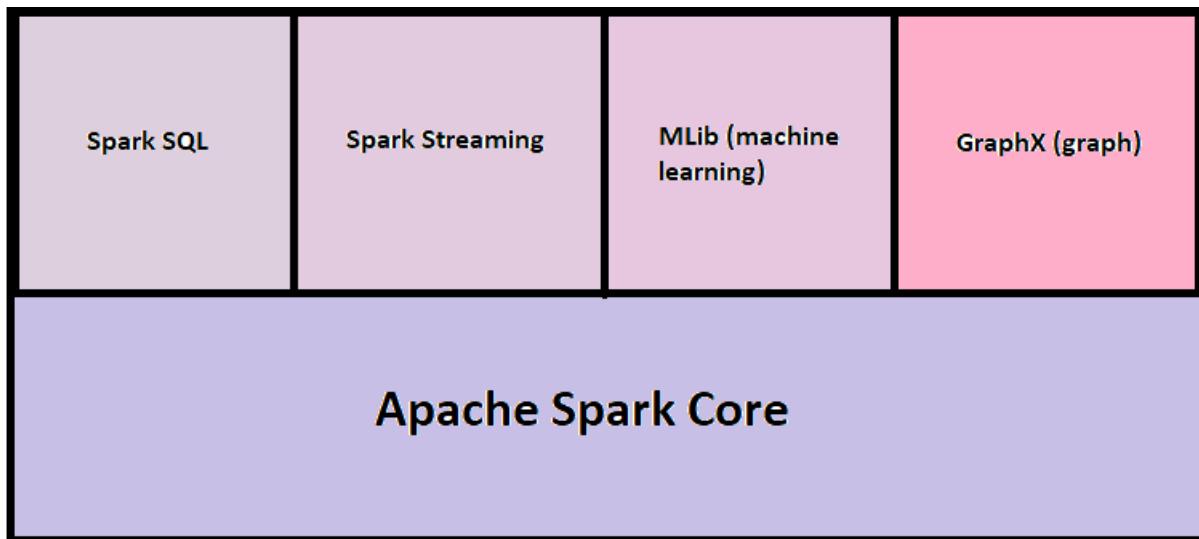


Figura 2.11: Componentes que conforman Apache Spark.

- **Spark Core:** Es el núcleo de Spark. Contiene librerías que se utilizan en todos los demás módulos.
- **Spark SQL:** Módulo para el procesamiento de datos estructurados y semi-estructurados. Esto se conoce como *SchemaRDD*. Este módulo hace posible el uso de lenguaje SQL para hacer consultas.
- **Spark Streaming:** Este módulo hace posible el procesamiento en tiempo real. Hace posible el flujo de gran cantidad de datos a alta velocidad.
- **MLlib:** Este módulo contiene herramientas y algoritmos muy variados para usar de manera fácil, práctica y escalable al *machine learning*.
- **GraphX:** Este módulo permite el procesamiento gráfico. No pinta gráficos, sino que realiza operaciones con grafos.

2.14. Interfaz Web

Una **interfaz de usuario** es el medio por el cual un usuario de un sistema computacional interactúa con el sistema. Se dice que una interfaz es **web** cuando el medio que despliega y renderiza la interfaz es un navegador web. Definir este concepto es importante ya que la interfaz de usuario de *Luminus* será una interfaz web.

CAPÍTULO 3

Evaluación y definición de requerimientos de la red distribuida

3.1. Descripción del prototipo

El prototipo actual busca definir el número de nodos y las características que estos deben tener para crear un ambiente de Big Data que nos permita realizar las pruebas a nuestro caso de estudio de manera satisfactoria pero que tampoco escape de las capacidades de computación con las que contamos.

Por otro lado busca definir como serán agrupados los datos que conforman el caso de estudio en los nodos que se establezcan. Para poder conocer con ello la distribución de los datos y el peso en memoria física que se le dará a cada nodo.

Se busca entonces encontrar características en común entre los datos que conforman el caso de estudio para de esta manera clasificarlos y facilitar su consulta al momento de aplicar algoritmos de minería de datos.

3.2. Análisis

3.2.1. Análisis de la red distribuida

Se pretende establecer las características que tendrá el cluster que se aplicará a la red distribuida para poder con esto, generarlo.

A continuación, se presenta el análisis de los factores que se consideran importantes para dicho objetivo:

- Se definió que es necesario realizar una replica de los datos en los nodos ya que si la información original no está disponible, en algún espacio en el tiempo se podrá utilizar la información de respaldo que se tiene de los mismos, con lo que se lograría que los datos sean mas persistentes.
- Para poder almacenar 2 copias de la misma información la original la replica, es necesario contar con al menos 2 nodos de datos y un nodo maestro.
- El nodo maestro no puede ser considerado nodo de datos ya que este cumple otra tarea, la cual es administrar y ordenar al resto de los nodos.
- También se definió que para manejar datos de la dimensión de 21GB (tamaño de la base de datos del caso de estudio), nodos de datos de menos de 2GB de capacidad de RAM no son eficientes e incluso llegan a tener problemas con las configuraciones de memoria que se realizan mas adelante, por lo que se considera que los nodos de datos deberán trabajar con al menos 2GB de RAM.
- Para poder comunicar el cluster dentro de una red local se utilizan 2 tecnologías Spark y Hadoop además de las dependencias de estas para funcionar como son SSH y Java. Por lo que es necesario considerar el espacio que ocupa la instalación de dichas tecnologías.

3.2.2. Análisis de los datos

La base de datos que se tiene planeado utilizar como caso de estudio tiene un total de 21GB de texto plano de información de productos que se venden en tiendas departamentales en la república mexicana.

Se trata de un archivo de datos de la PROFECO de uso libre el cual tiene los siguientes campos para cada producto.

producto , presentacion , marca , categoria , catalogo , precio , fecharegistro ,
cadenacomercial , giro , nombrecomercial , direccion , estado , municipio ,
latitud , longitud

El archivo contiene toda clase de productos comerciales desde alimentos, electrónica, linea blanca, papelería, etc. Cada producto viene acompañado de la información de la tienda departamental que lo comercializa. como ejemplo del contenido de dicho archivo se muestra los siguientes registros.

crayones caja 12 ceras. jumbo. c.b. 201423 crayola
material escolar utiles escolares 27.5 2011-05-18 00:00:00
abastecedora lumen papelerias abastecedora lumen sucursal villa coapa
cannes no. 6 esq. canal de miramontes distrito federal tlalpan
19.297 -99.1254
crayones caja 12 ceras. tamano regular c.b. 201034 crayola material escolar
utiles escolares 13.9 2011-05-18 00:00:00 abastecedora lumen papelerias
abastecedora lumen sucursal villa coapa cannes no. 6 esq. canal de miramontes
distrito federal tlalpan 19.297 -99.1254
galletas populares paquete 170 gr. marias gamesa galletas pastas y
harinas de trigo mercados 6.5 2011-01-10 00:00:00 walmart
tienda de autoservicio walmart boulevard bernardo quintana no.4113
esquina camino a sa queretaro santiago de queretaro 20.6162 -100.398
galletas populares paquete 170 gr. marias gamesa galletas pastas y
harinas de trigo mercados 6.6 2011-01-10 00:00:00 farmacia guadalajara
tienda de autoservicio farmacia guadalajara sucursal 326 ignacio picazo
no.25 norte casi esquina allende ponien tlaxcala chiautempan 19.3159 -98.1945

Estos ejemplos que fueron tomados estratégicamente para ser analizados como se lista a continuación.

- El archivo contiene información de varios productos que se venden en la misma tienda, conservando entonces la parte de datos referente a la tienda pero cambiando la parte del producto.
- El archivo contiene información del mismo producto vendido en varias tiendas departamentales en diferentes lugares, en esta caso, la información del producto es la misma mientras que la información de la tienda departamental cambia.
- El archivo contiene productos similares que se venden en la misma tienda o bien en otras tiendas que no son precisamente iguales pero que se pueden comparar entre ellos.

Usando el análisis formulado anteriormente se puede proponer diferentes alternativas de agrupación de los datos. Las que se consideraron se listan a continuación.

- Precio del producto
- Categoría del producto
- Tienda departamental

- Zona Geográfica
- Estado de la república donde se encuentra el producto

Por otro lado se determinó que el sistema de almacenamiento de datos que se va a utilizar en la red distribuida, será Hadoop, y revisando su modo de operación y de manejo de archivos en su HDFS este no permite que la forma en la que se agrupan los datos sea definida por el usuario por lo que, los datos serán distribuidos siguiendo la técnica de Hadoop.

Es más conveniente utilizar la forma en la que Hadoop distribuye los datos debido a que no es necesario controlar el acceso a los datos de manera manual indicando donde se encuentra cada uno de ellos, sino que Hadoop se encarga de esta tarea, además de ofrecer otros beneficios como la escalabilidad, la tolerancia a fallos, replicación en tiempo real, etc.

A pesar de esto, el análisis realizado para determinar los modos de agrupamiento más favorables no será desperdiciado pues este buscaba simplificar la operación de los algoritmos de minería de datos. y aunque no se aplique directamente sobre la distribución de los repositorios en los nodos, esta será aplicada al momento de ejecutar los algoritmos de minería de datos.

3.3. Diseño

3.3.1. Diseño de la red distribuida

Características de la red distribuida

Las características que tiene la red distribuida se enlistan a continuación: Se diseña una red distribuida que cuenta con 4 nodos en total.

Un nodo que cumple la función de nodo maestro.

Tres nodos que son utilizados como nodos de datos/replica.

Cada nodo usará la siguiente cantidad de memoria RAM:

Nodo Maestro: 6.7 GB.

Nodos Esclavos: 2 GB.

Se usa una conexión de red local para que los nodos puedan comunicarse.

Se asigna una IP estática a cada nodo para evitar problemas con la asignación dinámica de IPs mediante DHCP.

Cada nodo maestro y de datos/replica estará funcionando con base en un sistema operativo Ubuntu 16.04.

Se requiere que cada nodo de datos/replica cuente con 40 GB de almacenamiento en disco duro debido a que:

Se almacenarán 21 GB de información, y 21 GB de réplica en los nodos de información y de réplica respectivamente entre los 3 nodos.

Los programas que requieren tener en ejecución los nodos para funcionar correctamente, además de el sistema operativo los cuales ocupan 1.38GB de disco duro.

El espacio reservado libre para que Hadoop haga sus cambios y modificaciones de archivos en tiempo real como este lo requiera.

El espacio en disco libre que necesita el nodo para funcionar y seguir procesando datos.

Sólo los nodos de datos/replica almacenan y procesan información

El nodo maestro administra y ordena. Cada nodo funciona con Apache Spark 2.7 y Hadoop 3.1.1.

Red distribuida en la arquitectura del sistema

El diseño de la arquitectura del sistema y la red distribuida se muestran en la figura 3.1.

Como se puede observar en la arquitectura se tiene un nodo maestro y tres nodos de datos/replica conectados a través de Hadoop.

Se explicará como se busca que estos trabajen en conjunto una vez que todos los prototipos se encuentren terminados y como el diseño existente hasta este momento estará afectando dicho comportamiento.

El nodo maestro será el nodo donde el usuario experto instalará el ambiente para hacer uso de big data y configura sus

nodos, una vez que la instalación sea exitosa podrá ingresar sus datos, seleccionar los algoritmos que desea aplicar a los mismos y visualizará los resultados.

Mientras el usuario solicita operaciones directamente desde el nodo maestro los nodos de datos o replica se encontrarán accediendo a los datos que se encuentran alojados en cada uno de ellos y ejecutando las operaciones que se les soliciten sobre los mismos las solicitudes antes mencionadas serán hechas en su totalidad por el nodo maestro.

Posteriormente los resultados parciales obtenidos en cada uno de los nodos serán devueltos al nodo maestro para que este pueda generar el resultado final utilizando los resultados parciales de cada uno de los nodos y mostrarlo al usuario experto.

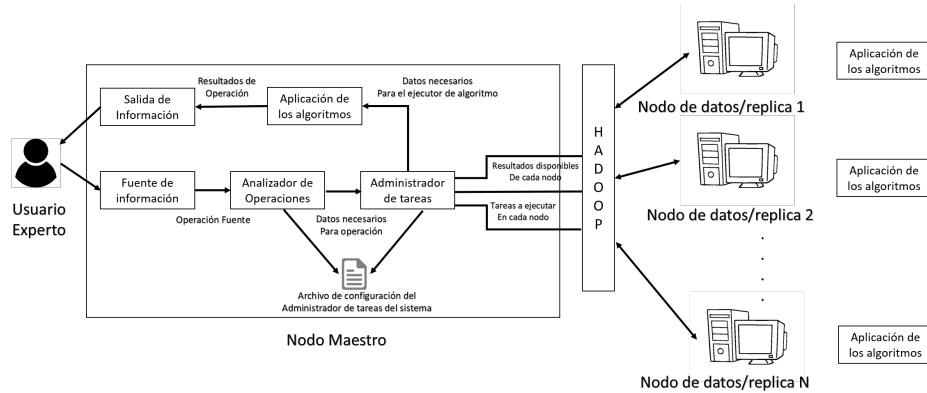


Figura 3.1: Arquitectura del sistema

3.3.2. Diseño de propuesta de agrupación de acuerdo a los nodos definidos en la red distribuida

Al usar una tecnología como Hadoop, este mismo es capaz de distribuir los datos sin que se diseñe una propuesta de agrupación por parte del programador sino que, Hadoop establece la propia.

Hadoop dentro de su arquitectura cuenta con un bloque conocido como HDFS en el cual es un sistema de archivos distribuido y tolerante a fallos. Funciona sobre el conjunto de los nodos de un cluster de Hadoop, balanceando la carga de archivos entre las máquinas del cluster, de forma equitativa. Gracias a su naturaleza distribuida, proporciona alta disponibilidad y altas prestaciones que le permiten ser capaz de manejar archivos de gran tamaño.

Para realizar esta tarea es necesario proporcionarle el archivo de datos completo a HDFS e indicarle el número de copias que se harán de los datos, el archivo proporcionado será segmentado en bloques de 128 MB por parte de Hadoop como se muestra en la figura 3.2.



Figura 3.2: Generación de bloques de un archivo

El sistema de Hadoop ingresará cada bloque considerando este bloque como bloque de datos en algún nodo y para las réplicas ingresara nuevamente el bloque considerando este bloque como bloque de replica 1, bloque de réplica 2 , bloque de réplica 3 y continua de la misma forma para la cantidad de bloques de réplica que se hayan indicado. Para cada una de las réplicas es indispensable que no sean asignadas en el mismo nodo que el bloque de datos o alguna otra réplica. No tiene sentido realizar más o igual número de réplicas que de nodos, debido a que las réplicas se realizan para garantizar el acceso a los bloques en todo momento, y si el bloque es accesible en un nodo y la réplica se encuentra en el mismo nunca sería utilizada.

Para ejemplificar la explicación anterior se mostrará la distribución que haría Hadoop para un sistema de 3 nodos con una réplica en la imagen 3.3.

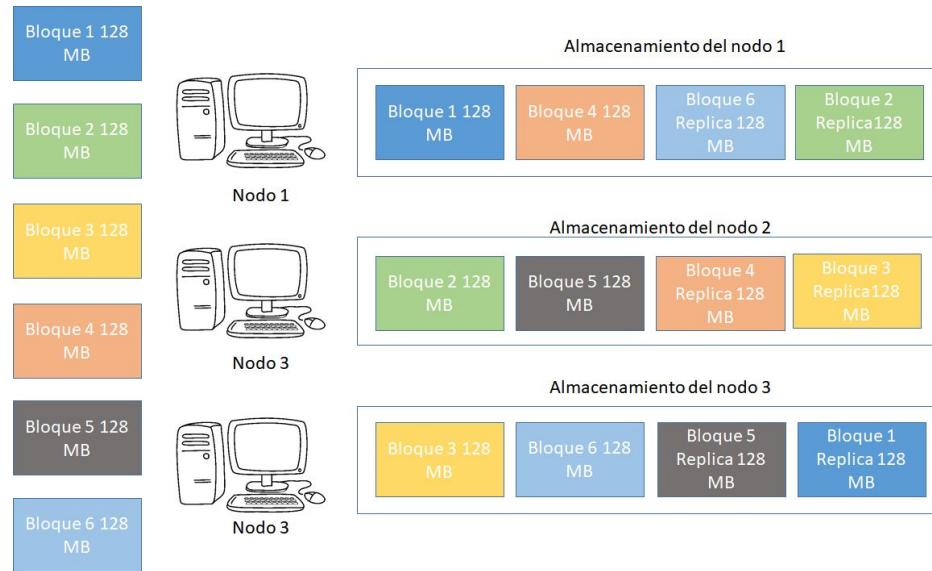


Figura 3.3: Generación de bloques de un archivo

Por lo que podemos ver que en caso de que alguno de los nodos deje de responder en los otros 2 nodos tenemos acceso a todos los datos, ya sea en el bloque de datos o en el bloque de réplica, hecho que no ocurriría en caso de que 2 nodos dejen de responder, el cual se solventa realizando 2 réplicas de los datos.

En tiempo de ejecución, para realizar su trabajo Hadoop siempre toma en cuenta primeramente los bloques de datos y en caso de no encontrarlos procede a hacer uso de las réplicas generadas de los mismos.

En el servidor maestro se guarda una información conocida como NameNode que permite conocer dónde se encuentran cada uno de los bloques y sus réplicas, lo que facilita sean encontradas dentro del cluster.

Con lo que podemos concluir que si bien no diseñamos la propuesta de agrupación de los datos comprendemos cómo es que esta funciona.

3.4. Desarrollo

Para que la red distribuida se encuentre en funcionamiento se siguió el Manual de Instalación de Luminus en sus secciones

1 Instalación de Apache Spark en el nodo

- 1.1. Instalación de la paquetería de java
- 1.2. Instalación de la paquetería de SSH
 - 1.2.1. Configuración
 - 1.2.2. Conexión
- 1.3. Instalación de Spark
 - 1.3.1. Configuración maestro

2. Instalación de Apache Spark en los nodos de datos

- 2.1. Instalación de la paquetería de java en los nodos de datos
- 2.2. Instalación de la paquetería de SSH en los nodos de datos
- 2.3. Instalación de Spark en los nodos de datos
 - 2.3.1. Configuración

3. Puesta en funcionamiento

- 3.1. Configuraciones para poner en funcionamiento Apache Spark en la red distribuida

Las cuales nos permitirán instalar Apache Spark para permitir la conexión entre los nodos de datos/replica y el maestro haciendo uso de una red de internet local en la que se encuentren conectados todos los nodos.

Además de contener todas las configuraciones necesarias para tal objetivo.

3.5. Pruebas

Para verificar que la instalación y configuración que se realizo al servidor Apache Spark es correcta será necesario aplicar un algoritmo de prueba.

Un algoritmo de prueba muy común es sparkPi el cual calcula el valor del número pi utilizando todos los nodos de la red distribuida.

Esto nos permitirá comprobar que existe conectividad dentro de la red y que se pueden realizar calculos con ella.

El comando a ejecutar es el siguiente

```
MASTER=spark:///[ IP del nodo maestro]:7077 run-example SparkPi
```

como se puede apreciar en la siguiente imagen

```
root@maestro:/home/maestro# MASTER=spark://192.168.1.88:7077 run-example SparkPiUsing Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
18/10/16 17:59:23 INFO SparkContext: Running Spark version 2.1.0
18/10/16 17:59:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
18/10/16 17:59:24 INFO SecurityManager: Changing view acls to: root
18/10/16 17:59:24 INFO SecurityManager: Changing modify acls to: root
18/10/16 17:59:24 INFO SecurityManager: Changing view acls groups to:
18/10/16 17:59:24 INFO SecurityManager: Changing modify acls groups to:
18/10/16 17:59:24 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root); groups with view permissions: Set()
; users with modify permissions: Set(root); groups with modify permissions: Set()
18/10/16 17:59:25 INFO Utils: Successfully started service 'sparkDriver' on port 46871.
18/10/16 17:59:25 INFO SparkEnv: Registering MapOutputTracker
18/10/16 17:59:25 INFO SparkEnv: Registering BlockManagerMaster
18/10/16 17:59:25 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
18/10/16 17:59:25 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
18/10/16 17:59:25 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-d7ab33be-238f-481a-a2ee-b9a030ea26f4
18/10/16 17:59:25 INFO MemoryStore: MemoryStore started with capacity 366.3 MB
18/10/16 17:59:25 INFO SparkEnv: Registering OutputCommitCoordinator
18/10/16 17:59:25 INFO Utils: Successfully started service 'SparkUI' on port 4040.
18/10/16 17:59:25 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://192.168.1.88:4040
18/10/16 17:59:25 INFO SparkContext: Added JAR file:/opt/spark/examples/jars/scott_2.11-
```

Figura 3.4: Ejecución de algoritmo SparkPi

cuando este termine podremos ver el resultado del valor de PI en la consola

```
90 bytes)
18/10/16 18:31:09 INFO TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, 192.168.1.88, executor 0, partition 1, PROCESS_LOCAL, 60
90 bytes)
18/10/16 18:31:10 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.1.88:41081 with 366.3 MB RAM, BlockManagerId(0, 1
92.168.1.88, 41081, None)
18/10/16 18:31:10 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor NettyRpcEndpointRef(null) (192.168.1.99:57620
) with ID 2
18/10/16 18:31:10 INFO BlockManager: Added broadcast_0_piece0 in memory on 192.168.1.88:41081 (size: 1172.0 B, free: 366.3 MB)
18/10/16 18:31:11 INFO BlockManagerMasterEndpoint: Registering block manager 192.168.1.99:44847 with 413.9 MB RAM, BlockManagerId(2, 1
92.168.1.99, 44847, None)
18/10/16 18:31:11 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1661 ms on 192.168.1.88 (executor 0) (1/2)
18/10/16 18:31:11 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor NettyRpcEndpointRef(null) (192.168.1.97:35414
) with ID 1
18/10/16 18:31:11 INFO TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 1486 ms on 192.168.1.88 (executor 0) (2/2)
18/10/16 18:31:11 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
18/10/16 18:31:11 INFO DAGScheduler: ResultStage 0 (reduce at SparkPi.scala:38) finished in 3.398 s
18/10/16 18:31:11 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 4.423339 s
Pi is roughly 3.141585709278546
18/10/16 18:31:11 INFO SparkUI: Stopped Spark web UI at http://192.168.1.88:4040
18/10/16 18:31:11 INFO StandaloneSchedulerBackend: Shutting down all executors
18/10/16 18:31:11 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
18/10/16 18:31:11 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/10/16 18:31:11 INFO MemoryStore: MemoryStore cleared
18/10/16 18:31:11 INFO BlockManager: BlockManager stopped
18/10/16 18:31:11 INFO BlockManagerMaster: BlockManagerMaster stopped
18/10/16 18:31:11 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
18/10/16 18:31:11 INFO SparkContext: Successfully stopped SparkContext
18/10/16 18:31:11 INFO ShutdownHookManager: Shutdown hook called
18/10/16 18:31:11 INFO ShutdownHookManager: Deleting directory /tmp/spark-6b3719b0-b845-424f-9115-be9cb0360bad
```

Figura 3.5: Valor de PI

cabe destacar que debido a que el valor es calculado en tiempo real con los recursos que se tiene en el cluster el valor puede variar cada ejecución pero será muy aproximado al valor real del número.

Ahora, si accedemos a la interfaz web de Spark podremos ver que un trabajo ha sido completado dentro del cluster,

antes de esta ejecución no se mostraba nada dentro de esta sección.

Spark Master at spark://maestro:7077

Spark 2.1.0

URL: spark://maestro:7077
 REST URL: spark://maestro:6066 (cluster mode)

Alive Workers: 4
 Cores in use: 7 Total, 0 Used
 Memory in use: 9.7 GB Total, 0.0 B Used
 Applications: 0 Running, 1 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20181016183058-192.168.1.88-46305	192.168.1.88:46305	ALIVE	4 (0 Used)	6.7 GB (0.0 B Used)
worker-20181016183059-192.168.1.97-40057	192.168.1.97:40057	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20181016183059-192.168.1.99-37930	192.168.1.99:37930	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20181016183105-192.168.1.98-33185	192.168.1.98:33185	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20181016183104-0000	Spark Pi	7	1024.0 MB	2018/10/16 18:31:04	root	FINISHED	7 s

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20181016183104-0000	Spark Pi	7	1024.0 MB	2018/10/16 18:31:04	root	FINISHED	7 s

Figura 3.6: Aplicación completada en Apache Spark

En caso de entrar a los detalles de la misma podremos observar que todos los trabajadores participaron en dicha aplicación y que se encuentran en el estado muerto debido a que la aplicación a finalizado su ejecución. así como también podemos visualizar los archivos de registro que generaron durante la ejecución de esta aplicación, entre otros detalles. otro punto importante es que también se puede consultar para cada uno de los nodos su participación

Application: Spark Pi

Spark 2.1.0

ID: app-20181016183104-0000
 Name: Spark Pi
 User: root
 Cores: Unlimited (7 granted)
 Executor Limit: Unlimited (4 granted)
 Executor Memory: 1024.0 MB
 Submit Date: Tue Oct 16 18:31:04 CDT 2018
 State: FINISHED

Executor Summary

ExecutorID	Worker	Cores	Memory	State	Logs
2	worker-20181016183059-192.168.1.99-37930	1	1024	KILLED	stdout stderr
1	worker-20181016183059-192.168.1.97-40057	1	1024	KILLED	stdout stderr
3	worker-20181016183105-192.168.1.98-33185	1	1024	KILLED	stdout stderr
0	worker-20181016183058-192.168.1.88-46305	4	1024	KILLED	stdout stderr

Removed Executors

ExecutorID	Worker	Cores	Memory	State	Logs
2	worker-20181016183059-192.168.1.99-37930	1	1024	KILLED	stdout stderr
1	worker-20181016183059-192.168.1.97-40057	1	1024	KILLED	stdout stderr
3	worker-20181016183105-192.168.1.98-33185	1	1024	KILLED	stdout stderr
0	worker-20181016183058-192.168.1.88-46305	4	1024	KILLED	stdout stderr

Figura 3.7: Detalles de la ejecución de la aplicación Spark Pi

en la ejecución de esta aplicación como se muestra en la figura 3.8 para el caso del nodo maestro

Spark 2.1.0 Spark Worker at 192.168.1.88:46305

ID: worker-20181016183058-192.168.1.88-46305
Master URL: spark://maestro:7077
Cores: 4 (0 Used)
Memory: 6.7 GB (0.0 B Used)

[Back to Master](#)

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

Finished Executors (1)

ExecutorID	Cores	State	Memory	Job Details	Logs
0	4	KILLED	1024.0 MB	ID: app-20181016183104-0000 Name: Spark Pi User: root	stdout stderr

Figura 3.8: Detalles de la ejecución de la aplicación Spark Pi en un nodo

Con lo que podemos concluir:

- El cluster funciona correctamente
- Existe comunicación entre los nodos
- Los nodos de datos/replica son capaces de identificar al nodo maestro y recibir instrucciones de el
- El nodo maestro es capaz de comunicar trabajos a los nodos de datos/replica y de interpretar los resultados de sus trabajos de manera satisfactoria

Por lo tanto, el prototipo uno concluye de manera exitosa

CAPÍTULO 4

Integración de requerimientos de la red distribuida

4.1. Descripción del prototipo

El prototipo actual busca que sea posible integrar los datos del caso de estudio a el cluster definido en el prototipo 1 que se encuentra en el capítulo [Evaluación y definición de requerimientos de la red distribuida](#).

Una vez que los datos sean cargados a los nodos, se busca comprobar que el cluster siga funcionando correctamente y que se puedan hacer consultas sobre los datos que este almacena, comprobando con esto su accesibilidad, disponibilidad y correcta asignación de los mismos a la red.

4.2. Análisis

4.2.1. Análisis de la adaptación de los datos a la red distribuida

El archivo de datos correspondiente al caso de estudio cuenta con 21GB de texto plano. Debido a que se definió que cada nodo de datos/replica cuenta con al menos 40GB de almacenamiento.Los datos podrán ser almacenados en la red distribuida sin causar problemas de almacenamiento ya que estos son soportados por las capacidades definidas en el prototipo 1 que fueron establecidas contemplando esta condición.

Para poder adaptar los datos del caso de estudio a la red distribuida creada con anterioridad se hará uso de Apache Hadoop.

Esto debido a que este cuenta con Hadoop Distributed File System (HDFS). HDFS es un sistema de archivos distribuido y tolerante a fallos. Funciona sobre el conjunto de los nodos de un cluster de Hadoop, balanceando la carga de archivos entre las máquinas del cluster, de forma equitativa. Gracias a su naturaleza distribuida, proporciona alta disponibilidad y altas prestaciones que le permiten ser capaz de manejar archivos de gran tamaño.

por lo que, haciendo uso de este sistema de manejo de archivos y teniendo las capacidades de almacenamiento en los nodos de la red distribuida es posible afirmar que se puede adaptar sin complicaciones el archivo de datos del caso de estudio.

4.3. Diseño

4.3.1. Diseño de la red distribuida con nodos de datos

La red distribuida cuenta con 3 nodos de datos/replica y un nodo maestro, una vez que se apliquen las configuraciones correspondientes a la asignación de los datos en los nodos de datos/replica así como un algoritmo para probar que los datos asignados son accesibles, la red distribuida deberá verse como la que se muestra en la imagen [Diseño de la red distribuida con nodos de datos](#)

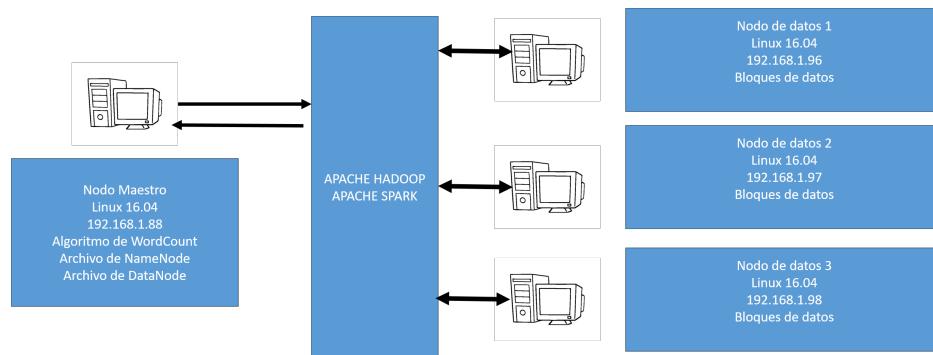


Figura 4.1: Diseño de la red distribuida con nodos de datos

la cual funciona conectando el cluster a una red local que les de una IP a cada uno de ellos la cual será definida como estática y se utilizará para las conexiones de manera distribuida tanto de Apache Hadoop como también de Apache Spark.

Los nodos de datos/replica no se comunicarán entre si al momento de ejecutar operaciones de computo en la red distribuida y para tal motivo utilizarán la conexión con el nodo maestro.

4.3.2. Diseño de pruebas de obtención de información

Para poder conocer si los datos fueron distribuidos correctamente y el cluster se encuentra en funcionamiento es posible utilizar un algoritmo que use los datos que se encuentran en el cluster y al final regrese un resultado.

Los algoritmos de big data que serán necesarios para este trabajo son algoritmos que utilizan como base una tecnología llamada Map Reduce.

Se encontró que existen algoritmos de prueba dentro de Hadoop que pueden ser utilizados para comprobar el correcto funcionamiento de la red. sin embargo, se busca comprender como es que estas pruebas funcionan.

Y a su vez buscar que el ejemplo que sea aplicado sobre los datos del caso de estudio se adapte mejor a estos y ofrezca resultados mas interesantes.

WordCount tradicional

La prueba que se va a realizar es el Contador de palabras.^º "WordCount." este algoritmo lo que hace es contar todas las palabras que se encuentran dentro de un archivo y decir cuantas coincidencias de la misma palabra se encontraron. El algoritmo funciona de la siguiente manera:

- Cada que encuentra una nueva palabra la agrega al listado de palabras con el valor de 1 que significa que solo ha sido encontrada una vez.
- En caso de que la palabra sea encontrada nuevamente, entonces se reemplazara el número asociado a esa palabra por el número que tenia en ese momento + 1.
- Termina cuando llega al final del archivo y por lo tanto todas las palabras nuevas fueron registradas y se conoce cuantas veces aparecen en el archivo.

El procedimiento descrito anteriormente se explicará de igual forma con el diagrama de flujo que se muestra en la Imagen [Diagrama de flujo del algoritmo contador de palabras](#).

Los pasos descritos anteriormente, son los pasos que serían utilizados si se tratara de un algoritmo que se ejecuta sin el uso de Map Reduce.

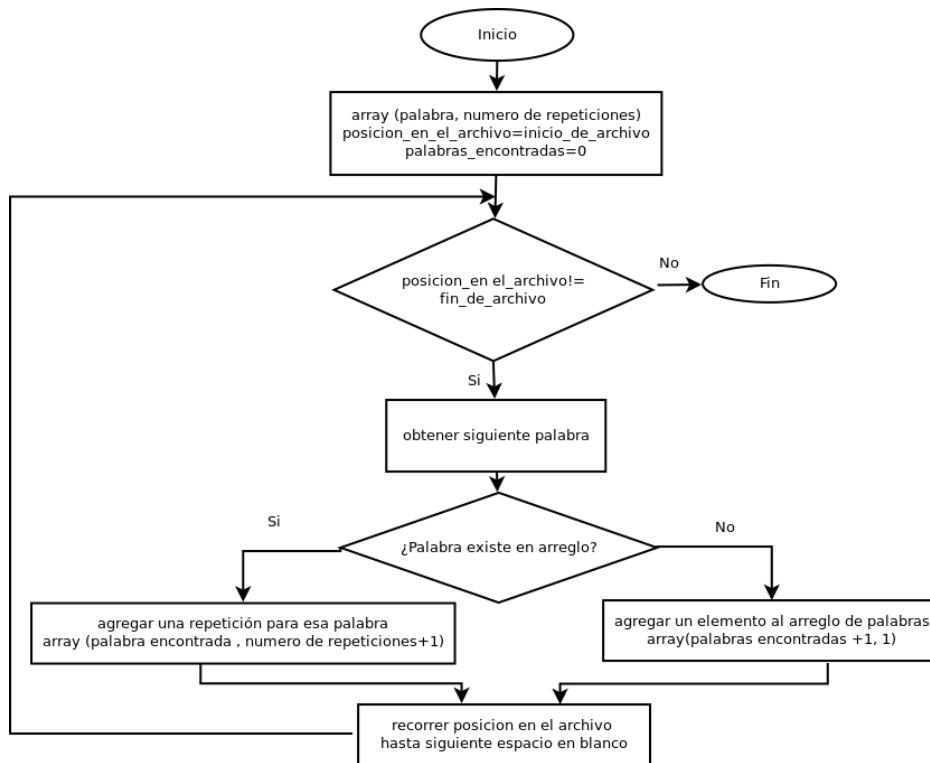


Figura 4.2: Diagrama de flujo del algoritmo contador de palabras

WordCount con el uso de Map Reduce

Map Reduce es una técnica que descompone un trabajo grande en tareas individuales las cuales pueden ser ejecutadas por separado en diferentes computadoras que componen un cluster, y estas al final pueden unir sus resultados individuales para calcular los resultados finales.

Función Map: Toma un conjunto de datos y los convierte en otro conjunto de datos, donde los elementos individuales se dividen en entradas del nuevo conjunto con la estructura (clave-valor).

de tal forma que lo que una función Map en este ejemplo haría sería convertir el conjunto de palabras en un conjunto diferente donde: Clave: toma el valor de la palabra que se encontró en el archivo Valor: asigna el valor de 1 a otras las palabras encontradas en el archivo. Veamos el siguiente conjunto de entrada dentro del archivo:

botella , refresco , Botella , BOTELLA, Fresco , Refresco , botellarefresco .

para este archivo el conjunto de salida de la función map, seria:

(botella ,1) , (refresco ,1) , (Botella ,1) , (BOTELLA,1) ,
(Fresco ,1) ,(Refresco ,1) , (botellarefresco ,1) .

Función Reduce: toma la salida del mapa y combina las entradas para generar un conjunto mas pequeño de datos De tal forma que lo que la función reduce haría en este ejemplo sería buscar las palabras que sean las mismas de las entregadas por cada nodo y agruparlas. veamos el mismo ejemplo, suponiendo que el trabajo fue realizado por 3 nodos de datos, veamos lo que haría la función reduce.

nodo1
(botella ,1) , (refresco ,1) .
nodo2
(Botella ,1) , (BOTELLA,1) .
nodo3
(Fresco ,1) ,(Refresco ,1) , (botellarefresco ,1) .

para estas entradas, el conjunto de salida de la función reduce seria:

(BOTELLA,3) , (REFresco,3) , (BOTELLAREFRESCO ,1) .

Sin embargo, cabe destacar que el sistema de Hadoop no solo trabaja con las funciones map y reduce, sino que tiene otras funciones internas que el sistema controla.

En la imagen [Funcionamiento completo de Hadoop, con todas sus operaciones](#) se puede visualizar el funcionamiento completo que tendría que ejecutarse en el cluster para llevar a cabo este algoritmo. Teniendo como referencia la imagen

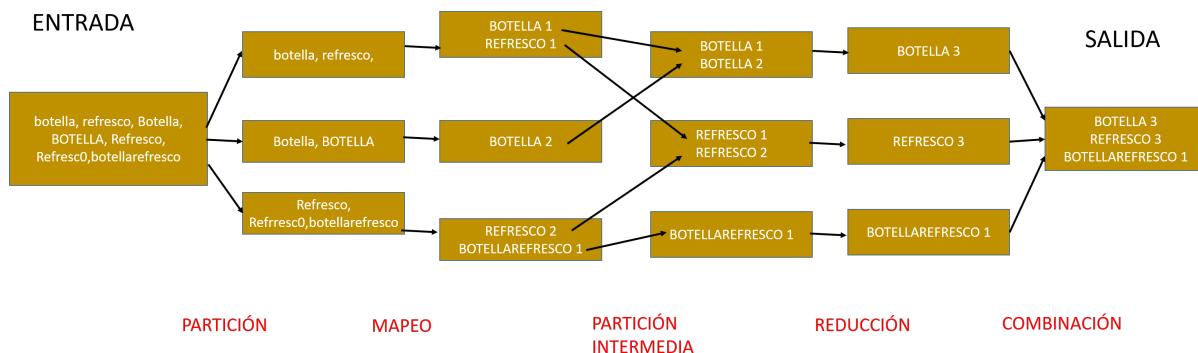


Figura 4.3: Funcionamiento completo de Hadoop, con todas sus operaciones

[Funcionamiento completo de Hadoop, con todas sus operaciones](#) se explicarán los módulos que ejecuta Hadoop de forma interna para una aplicación Map Reduce de forma breve.

- Partición:** Se requiere definir un parámetro de partición, esto para que se puedan reconocer las unidades a analizar y con esto poder asignar tareas a los nodos. el parámetro de partición puede ser cualquier cosa, por ejemplo, dividir por espacio, coma, punto y coma, o incluso por una nueva línea ('n').
- Mapeo:** Se explico anteriormente en [WordCount con el uso de Map Reduce Función Map](#).
- Partición intermedia:** Se busca generar grupos con los datos que después de pasar por el modulo de mapeo y tener la estructura de salida de este modulo tienen la misma CLAVE, al cumplir esta condición se asignan en el mismo grupo. Se generan tantos grupos como claves existan.
- Reducción:** Se explico anteriormente en [WordCount con el uso de Map Reduce Función Reduce](#).
- Combinación:** Todos los datos de salida que arroja la función de reducción son combinados y puestos en un solo grupo para generar un resultado final.

4.4. Desarrollo

Para que Instalar Apache Hadoop en la red distribuida en funcionamiento se siguió el Manual de Instalación de Luminus en sus secciones

- Instalación de Apache Hadoop en el nodo maestro
 - Instalación de Hadoop
 - Configuración
 - Archivos de configuración
- Instalación de Apache Hadoop en los nodos de datos
 - Instalación de Hadoop en los nodos de datos
 - Configuración
- Puesta en funcionamiento
 - Puesta en funcionamiento del Cluster manejado por el Servidor Apache Hadoop

Las cuales nos permitirán instalar Apache Hadoop el cual tendrá al mismo tiempo todas las configuraciones necesarias requeridas para este proyecto, tanto en el nodo maestro como también en los nodos de datos/replica. Dentro del manual de instalación se explica para cada uno de los archivos de configuración de hadoop como es que estos se configuran y cual es el objetivo de cada configuración de manera detallada. Una vez que las configuraciones se hayan ejecutado de manera correcta y el cluster se encuentre en funcionamiento haciendo uso de hadoop, se subirá un archivo a la plataforma, esto se hará siguiendo el manual de instalación de luminus en la sección

6.2. Subir un archivo al HDFS

En el momento de subir el archivo al HDFS se puede comprobar que la red distribuida permite realizar esta operación para lo cual es indispensable comunicarse con todos los nodos que se encuentran dentro de la misma. Esto con el objetivo de realizar el almacenamiento de manera distribuida haciendo uso de todos los nodos de datos/replica. Por lo tanto se sabe que la red distribuida tiene conectividad y se encuentra funcionando de manera apropiada.

4.4.1. Algoritmo de prueba

El siguiente algoritmo escrito en JAVA servirá para comprobar que se pueden realizar operaciones sobre los datos que se encuentran en el cluster.

Como se explico en la sección [Diseño de pruebas de obtención de información](#) de este capítulo, se utilizará el algoritmo wordcount para cumplir con este objetivo. También, se menciono que las únicas secciones que se requiere programar para ejecutar un algoritmo de tipo map reduce es la función map y la función reduce. ya que hadoop se encarga de el resto de funciones.

Por otro lado se explico la forma en que este algoritmo en particular funcionar y un ejemplo de su funcionamiento en la imagen [Funcionamiento completo de Hadoop, con todas sus operaciones](#).

Se decidió que se considere que se encontró una palabra cada que aparezca una "," dentro del archivo como parámetro de la función de partición, esto debido a que cada atributo de la tabla de datos esta separado por una coma y esto nos permitiría comprobar cuantas veces aparece un atributo dentro del archivo en lugar de solamente conocer la repetición de las palabras por separado. Lo cual esta orientado para el archivo de datos del caso de estudio y nos puede dar información de la frecuencia con la que ciertos registros aparecen dentro del mismo.

El código JAVA para tal objetivo se puede visualizar en el [Anexo A: Código JAVA Map-Reduce](#) dentro del cual se pueden destacar algunas observaciones

- Se importan las librerías de Apache Hadoop para que pueda reconocerse que se trata de un programa Map Reduce que se ejecutará sobre este programa.
- Se tiene la clase main la cual principalmente manda a llamar a las clases map y reduce, establece los valores por defecto para empezar a contar, Establece los canales de lectura y escritura para que este pueda acceder a los archivos del cluster a analizar y pueda tambien escribir sus resultados.
- Se tiene la clase map la cual toma la palabra encontrada y le da la estructura (CLAVE, VALOR) , además de pasarl a mayúsculas para que considere que se trata de la misma palabra cuando tenga las mismas letras sin importar si originalmente estaba escrita en mayúsculas, minúsculas o una combinación de ambas.
- La clase reduce que cuenta cuantas veces se repiten en total las palabras en el grupo que se genero de todas las palabras iguales que se encontraron en cada uno de los nodos. para sacar una suma total para cada palabra del archivo.

Este algoritmo tiene que ser compilado ya sea con un IDE de java o directamente desde Hadoop para generar el jar que posteriormente sera ejecutado por Hadoop.

El código desde consola para generar este JAR es el siguiente.

```
bin/hadoop com.sun.tools.javac.Main [clase_principal_java].java  
jar cf [Nombre_jar].jar [clase_principal_java]*.class
```

4.5. Pruebas

Una vez que se tiene el .JAR a ejecutar se puede hacer uso del siguiente comando para que este entre en funcionamiento dentro de la red distribuida.

```
yarn jar <ruta/a/archivo.jar> <NombreDeClase> <Parametros>
```

Para nuestro ejemplo específico el comando seria:

```
root@maestro:/opt/hadoop/etc/hadoop: yarn jar /home/mayra/Escritorio/MRProgramsDemo.jar  
PackageDemo.WordCount "user/root/productos/inserts.txt" output6
```

output6 sera una carpeta que se creará en el sistema de archivos de HDFS para almacenar los resultados de salida. Esta carpeta puede tener el nombre que se desee y se asigna en esta sección, sin embargo, no puede existir dentro del sistema de archivos al momento de ejecutar este comando. La carpeta se creará en el directorio /user/root/nombreasignado Una vez que se ejecute esta instrucción se procederá a revisar las conexiones con los nodos y hacer los ajustes necesarios para comenzar a ejecutar este algoritmo. esta información de salida y el comienzo de la ejecución del algoritmo con la parte de la función map pueden verse en la imagen [Ejecución de algoritmo map reduce para contar palabras: Comenzando el proceso](#) este algoritmo. esto se hará para todos los porcentajes de la función map, una vez que estos finalicen, comenzará a ejecutar la función reduce y con ella sus porcentajes de progreso como se muestra en la imagen [Ejecución de algoritmo map reduce para contar palabras: Inicio de funcion reduce](#).

```

nodo3:38667          RUNNING      nodo3:8042
          0
nodo2:44891          RUNNING      nodo2:8042
          0
root@maestro:/opt/hadoop/etc/hadoop# yarn jar /home/maestro/Escritorio/MRProgramsDemo.jar PackageDemo.WordCount /user/luminus/datosproductos/all_data.csv /user/root/output6
2018-10-18 03:20:28,566 INFO client.RMProxy: Connecting to ResourceManager at maestro/192.168.1.88:8032
2018-10-18 03:20:29,501 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1539850756835_0001
2018-10-18 03:20:32,024 INFO input.FileInputFormat: Total input files to process : 1
2018-10-18 03:20:33,574 INFO mapreduce.JobSubmitter: number of splits:154
2018-10-18 03:20:33,791 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
2018-10-18 03:20:34,547 INFO mapreduce.JobSubmitter: Submitting tokens for job : job_1539850756835_0001
2018-10-18 03:20:34,554 INFO mapreduce.JobSubmitter: Executing with tokens: []
2018-10-18 03:20:34,917 INFO conf.Configuration: resource-types.xml not found
2018-10-18 03:20:34,918 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2018-10-18 03:20:35,281 INFO impl.YarnClientImpl: Submitted application application_1539850756835_0001
2018-10-18 03:20:35,328 INFO mapreduce.Job: The url to track the job: http://maestro:8088/proxy/application_1539850756835_0001/
2018-10-18 03:20:35,329 INFO mapreduce.Job: Running job: job_1539850756835_0001
2018-10-18 03:20:50,683 INFO mapreduce.Job: Job job_1539850756835_0001 running in uber mode : false
2018-10-18 03:20:50,694 INFO mapreduce.Job: map 0% reduce 0%
2018-10-18 03:21:21,278 INFO mapreduce.Job: map 1% reduce 0%
2018-10-18 03:21:27,533 INFO mapreduce.Job: map 2% reduce 0%
2018-10-18 03:21:50,770 INFO mapreduce.Job: map 3% reduce 0%
2018-10-18 03:22:20,359 INFO mapreduce.Job: map 4% reduce 0%
2018-10-18 03:22:40,513 INFO mapreduce.Job: map 5% reduce 0%
2018-10-18 03:22:53,605 INFO mapreduce.Job: map 6% reduce 0%
2018-10-18 03:23:50,094 INFO mapreduce.Job: map 7% reduce 0%
2018-10-18 03:23:56,136 INFO mapreduce.Job: map 8% reduce 0%
2018-10-18 03:24:33,320 INFO mapreduce.Job: map 9% reduce 0%
2018-10-18 03:24:52,391 INFO mapreduce.Job: map 10% reduce 0%

```

Figura 4.4: Ejecución de algoritmo map reduce para contar palabras: Comenzando el proceso

```

2018-10-18 04:42:43,728 INFO mapreduce.Job: map 92% reduce 0%
2018-10-18 04:43:05,769 INFO mapreduce.Job: map 93% reduce 0%
2018-10-18 04:43:40,041 INFO mapreduce.Job: map 94% reduce 0%
2018-10-18 04:43:50,060 INFO mapreduce.Job: map 95% reduce 0%
2018-10-18 04:44:43,179 INFO mapreduce.Job: map 96% reduce 0%
2018-10-18 04:44:50,190 INFO mapreduce.Job: map 97% reduce 0%
2018-10-18 04:45:32,416 INFO mapreduce.Job: map 98% reduce 0%
2018-10-18 04:46:29,550 INFO mapreduce.Job: map 99% reduce 0%
2018-10-18 04:46:35,560 INFO mapreduce.Job: map 100% reduce 0%
2018-10-18 04:47:37,928 INFO mapreduce.Job: map 100% reduce 1%
2018-10-18 04:47:50,952 INFO mapreduce.Job: map 100% reduce 2%
2018-10-18 04:48:24,094 INFO mapreduce.Job: map 100% reduce 3%
2018-10-18 04:49:11,337 INFO mapreduce.Job: map 100% reduce 4%
2018-10-18 04:49:35,550 INFO mapreduce.Job: map 100% reduce 5%
2018-10-18 04:50:24,729 INFO mapreduce.Job: map 100% reduce 6%
2018-10-18 04:50:54,842 INFO mapreduce.Job: map 100% reduce 7%
2018-10-18 04:51:43,186 INFO mapreduce.Job: map 100% reduce 8%
2018-10-18 04:52:19,329 INFO mapreduce.Job: map 100% reduce 9%
2018-10-18 04:53:01,520 INFO mapreduce.Job: map 100% reduce 10%
2018-10-18 04:53:46,821 INFO mapreduce.Job: map 100% reduce 11%
2018-10-18 04:54:48,095 INFO mapreduce.Job: map 100% reduce 12%
2018-10-18 04:55:19,227 INFO mapreduce.Job: map 100% reduce 13%
2018-10-18 04:56:33,635 INFO mapreduce.Job: map 100% reduce 14%
2018-10-18 04:57:03,730 INFO mapreduce.Job: map 100% reduce 15%
2018-10-18 04:58:16,120 INFO mapreduce.Job: map 100% reduce 16%
2018-10-18 04:58:58,266 INFO mapreduce.Job: map 100% reduce 17%
2018-10-18 04:59:52,628 INFO mapreduce.Job: map 100% reduce 18%
2018-10-18 05:00:40,852 INFO mapreduce.Job: map 100% reduce 19%
2018-10-18 05:01:41,202 INFO mapreduce.Job: map 100% reduce 20%
2018-10-18 05:02:24,356 INFO mapreduce.Job: map 100% reduce 21%
2018-10-18 05:03:06,483 INFO mapreduce.Job: map 100% reduce 22%
2018-10-18 05:03:50,788 INFO mapreduce.Job: map 100% reduce 23%
2018-10-18 05:04:32,926 INFO mapreduce.Job: map 100% reduce 24%
2018-10-18 05:05:39,345 INFO mapreduce.Job: map 100% reduce 25%
2018-10-18 05:06:21,460 INFO mapreduce.Job: map 100% reduce 26%
2018-10-18 05:07:15,633 INFO mapreduce.Job: map 100% reduce 27%
2018-10-18 05:08:16,934 INFO mapreduce.Job: map 100% reduce 28%
2018-10-18 05:08:47,024 INFO mapreduce.Job: map 100% reduce 29%
2018-10-18 05:09:53,429 INFO mapreduce.Job: map 100% reduce 30%
2018-10-18 05:10:41,565 INFO mapreduce.Job: map 100% reduce 31%

```

Figura 4.5: Ejecución de algoritmo map reduce para contar palabras: Inicio de función reduce

Cuando se complete la función reduce procederá a notificar que el algoritmo fue ejecutado correctamente y mostrará estadísticas de:

- File System Counter
- Job Counter
- MapReduce Framework

Dichas estadísticas y resultados finales para este algoritmo se muestran en las figuras [Ejecución de algoritmo map reduce para contar palabras: Termina la ejecución del algoritmo](#) y [Ejecución de algoritmo map reduce para contar palabras: Estadísticas](#)

```

2018-10-18 05:42:11,402 INFO mapreduce.Job: map 100% reduce 99%
2018-10-18 05:42:29,432 INFO mapreduce.Job: map 100% reduce 100%
2018-10-18 05:43:25,002 INFO mapreduce.Job: Job job_1539850756835_0001 completed successfully
2018-10-18 05:43:27,429 INFO mapreduce.Job: Counters: 57
    File System Counters
        FILE: Number of bytes read=77218820448
        FILE: Number of bytes written=103931282332
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=20664348064
        HDFS: Number of bytes written=212338004
        HDFS: Number of read operations=467
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Killed map tasks=23
        Killed reduce tasks=1
        Launched map tasks=176
        Launched reduce tasks=2
        Other local map tasks=21
        Data-local map tasks=154
        Rack-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=16923807
        Total time spent by all reduces in occupied slots (ms)=1447493
0
        Total time spent by all map tasks (ms)=16923807
        Total time spent by all reduce tasks (ms)=7237465
        Total vcore-milliseconds taken by all map tasks=16923807
        Total vcore-milliseconds taken by all reduce tasks=7237465
        Total megabyte-milliseconds taken by all map tasks=11542036374
        Total megabyte-milliseconds taken by all reduce tasks=98719022
60
    Map-Reduce Framework
        Map input records=62530716
        Map output records=1015403483
        Map output bytes=24648238567
        Map output materialized bytes=26679114886
        Input split bytes=19250
        Combine input records=0
        Combine output records=0

```

Figura 4.6: Ejecución de algoritmo map reduce para contar palabras: Termina la ejecución del algoritmo

```

50
Map-Reduce Framework
  Map input records=62530716
  Map output records=1015403483
  Map output bytes=24648238567
  Map output materialized bytes=26679114886
  Input split bytes=19250
  Combine input records=0
  Combine output records=0
  Reduce input groups=7676839
  Reduce shuffle bytes=26679114886
  Reduce input records=1015403483
  Reduce output records=7676839
  Spilled Records=3946856040
  Shuffled Maps =154
  Failed Shuffles=0
  Merged Map outputs=154
  GC time elapsed (ms)=135866
  CPU time spent (ms)=5140450
  Physical memory (bytes) snapshot=63333994496
  Virtual memory (bytes) snapshot=358570815488
  Total committed heap usage (bytes)=50046279680
  Peak Map Physical memory (bytes)=435355648
  Peak Map Virtual memory (bytes)=2311077888
  Peak Reduce Physical memory (bytes)=1119645696
  Peak Reduce Virtual memory (bytes)=2891509760
Shuffle Errors
  BAD_ID=0

```

Figura 4.7: Ejecución de algoritmo map reduce para contar palabras: Estadísticas

Ahora, para poder visualizar el resultado de este algoritmo se puede hacer un listado de los archivos que contiene el directorio de salida que se creo, para ello se utiliza el comando siguiente

```
root@maestro:/opt/hadoop/etc/hadoop# hdfs dfs -ls /user/root/output6
```

Con lo cual se genera la salida que se despliega en la imagen 4.8 la cual muestra que la operación fue exitosa y las dimensiones generadas para el archivo de salida.

```

root@maestro:/opt/hadoop/etc/hadoop# hdfs dfs -ls /user/root/output6
Found 2 items
-rw-r--r--  2 root supergroup          0 2018-10-18 05:42 /user/root/output6/_SUCCESS
-rw-r--r--  2 root supergroup  212338004 2018-10-18 05:42 /user/root/output6/part-r-00000

```

Figura 4.8: Contenido del directorio de salida

Debido a que se trata de un archivo muy grande, para efectos de simplificación solo se mostrará en este documento un fragmento de dicho archivo que deje ver el trabajo que fue realizado dicho fragmento se muestra en la imagen 4.9. Dentro de esta imagen podemos ver por ejemplo, cuantas veces aparecen determinados precios dentro del archivo, direcciones o bien descripciones de productos, entre otros.

Por lo cual se puede ver que el archivo fue estudiado en su totalidad y que se tienen coincidencias para diferentes entradas que se encontraban dentro del archivo.

Un resultado interesante obtenido que se puede observar en la imagen 4.10 es que, al menos para este archivo la fecha de registro es irrelevante pues para cada articulo se tiene una diferente o bien comparten una misma fecha cuando la hora establecida es 0hrs para una gran cantidad de registros que fueron dados de alta el mismo día, pero en realidad

part-r-00000 (~/Vídeos) - gedit

Abrir ▾ Guardar

```
"1996.65"      20
"19963/ 19934 ASST. MARVEL. SUPER HERO SQUAD. QUINJET"  11
"1997.00"      10
"19977. BEYBLADE. MOBILE BEYSTADIUM"      25
"19979.10"      2
"1998.00"      152
"19980. BEYBLADE. SUPER VORTEX BATTLE SET"      8
"19987. TONKA CHUCK & FRIENDS. ROWDY EL CAMION DE BASURA"      171
"19988.00"      2
"1999.00"      1272
"1999.20"      66
"1999.50"      6
"19990.00"      78
"19998.00"      22
"19998.30"      2
"19999.00"      88
"19999.20"      90
"10. DE MAYO NO. 2 20 DE NOVIEMBRE ESQ. PORFIRIO DIAZ"  24
"10. DE MAYO 200      38266
"10. DE MAYO MZ-C 24-B ESQ. TENANGO DEL VALLE      59318
"10. DE MAYO NO. 18 ESQUINA ALLENDE      30
"10. DE MAYO NO. 18 ESQUINA ALLENDE"      2295
"2 CAJAS CON 14 CAPSULAS DE 20 MG. C/U" 30747
"2 CAJAS CON 30 TABLETAS C/U DE 10 MG." 28985
"2 CAJAS CON 30 TABLETAS C/U DE 20 MG." 28407
"2 CAJAS CON 8 COMPRIMIDOS C/U DE 400 MG.-100 MG."      30791
"2 FCOS AMP DE 20 MG."  11
"2 ORIENTE NO. 225      122
"2 ORIENTE NO. 225"      548
"2 PLIEGOS"      1225
"2 PLIEGOS. CARTULINA"  3771
"2 PONIENTE 704 3
"2 PTE. NO. 704 187
"2.00"      248
"2.03"      2
"2.05"      18
```

Figura 4.9: Contenido del archivo de resultados

no proporciona información real de los artículos y genera mucho ruido en el archivo.

Existen más entradas de fechas que información de los productos, comportamiento que dificultaría el análisis de los datos en un futuro por lo cual este atributo será retirado del archivo correspondiente al caso de estudio. Con lo que

"2015-09-23 17:30:10.397"	1
"2015-09-23 17:30:12.187"	1
"2015-09-23 17:30:15.047"	1
"2015-09-24 00:00:00.000"	87671
"2015-09-25 00:00:00.000"	73318
"2015-09-25 09:36:52.837"	1
"2015-09-25 12:49:06.920"	1
"2015-09-25 12:49:07.767"	1
"2015-09-25 12:49:14.497"	1
"2015-09-25 12:49:28.077"	1
"2015-09-25 12:49:29.027"	1
"2015-09-25 12:49:52.087"	1
"2015-09-25 12:50:07.230"	1
"2015-09-25 12:51:17.713"	1
"2015-09-25 12:51:22.007"	1
"2015-09-25 12:51:23.080"	1
"2015-09-25 12:51:26.690"	1
"2015-09-25 12:52:03.880"	1
"2015-09-25 12:52:28.507"	1
"2015-09-25 12:52:29.143"	1
"2015-09-25 12:53:02.143"	1
"2015-09-25 13:15:48.707"	1
"2015-09-25 13:45:30.567"	1
"2015-09-25 14:48:43.067"	1
"2015-09-25 14:50:48.520"	1
"2015-09-25 14:50:49.510"	1
"2015-09-25 14:50:50.840"	1
"2015-09-25 15:00:21.033"	1
"2015-09-25 15:03:53.300"	1
"2015-09-25 15:47:18.000"	1

Figura 4.10: Fechas de registro en el archivo de resultados

podemos concluir:

- El cluster funciona correctamente
- Existe comunicación entre los nodos
- Los nodos de datos/replica son capaces de identificar al nodo maestro y recibir instrucciones de él
- El nodo maestro es capaz de comunicar trabajos a los nodos de datos/replica y de interpretar los resultados de sus trabajos de manera satisfactoria
- Se tiene el archivo del caso de estudio almacenado de manera distribuida en los nodos
- El archivo del caso de estudio es accesible y se pueden ejecutar operaciones sobre él

Por lo tanto, el prototipo dos concluye de manera exitosa

CAPÍTULO 5

Instalador de Luminus

5.1. Descripción del prototipo

Uno de los objetivos de la realización de este trabajo terminal es permitir al usuario empezar a hacer uso de Big Data de una manera sencilla y sin demasiadas complicaciones. Por lo que se llevo a cabo el desarrollo de un instalador que simplifique el proceso de puesta en marcha del ambiente de análisis de datos.

Se trata de una herramienta que asista al usuario en disminuir el número de pasos necesarios para instalar un ambiente que permita llevar a cabo la realización de Big Data sobre de el.

Podría decirse que la utilidad de este prototipo se refleja al comparar el número de pasos que se enuncian en el manual de instalación de *Luminus*, contra el número de pasos que se siguen al utilizar el instalador mismos que serán comparados dentro de esta sección.

5.2. Análisis

La manera en que se pretende que el instalador simplifique el proceso de instalación es automatizando algunas tareas que de otra forma el usuario tendría que realizar una a una, existiendo así la posibilidad de que éste mismo cometa algún error u omita algún paso, y por lo tanto, el ambiente de análisis de datos no pueda ponerse en funcionamiento. Existen diferentes tecnologías para solventar la instalación y configuración de un conjunto previamente determinado de tecnologías, por ejemplo, Docker.

Sin embargo, para el uso de Big Data se utilizan tecnologías como Apache Hadoop y Apache Spark, mismas que para estas tecnologías, la instalación y puesta en marcha con Docker se encuentran en fase experimental lo que significa que no es consistente y generaría muchos problemas al momento de querer utilizarlo [18].

Para solventar esto, se decidió hacer uso del Shell Script de Unix/Linux el cual es simplemente un programa que lee los comandos que se teclean y los convierte en una forma mas entendible para el sistema Unix/Linux. También incluye algunas sentencias básicas de programación que permiten: tomar decisiones, realizar ciclos y almacenar valores en variables [19].

Por lo que, dadas las características de Shell a pesar de que este es la forma mas rudimentaria de realizar esta tarea, es la que de acuerdo a nuestra investigación ofrece mejores resultados para las tecnologías involucradas.

Utilizar Shell Script por otro lado, implica un conjunto de complicaciones ya que todo lo que el script haga, tendrá que ser programado, y esto implica un gran número de validaciones que de utilizar otras tecnologías no se tendrían que considerar, ya que estas serían solventadas por la propia tecnología y su robustez.

5.3. Diseño

La funcionalidad completa del instalador originalmente estaba planteada para contemplar todos los prototipos del Trabajo Terminal, sin embargo, al cambiar la definición del funcionamiento que tendrían los prototipos 3,4 y 5 se encontró que incluirlos en el instalador, no sería lo mas apropiado de acuerdo a la manera en la que estos funcionan y como esta es independiente en cierto sentido del instalador.

Inicialmente, cuando se plantearon estos prototipos de manera general su funcionamiento en conjunto estaba pensado para ser incorporado como un sitio web, por lo que el instalador podría poner los archivos correspondientes de funcionamiento en las carpetas destinadas para el sitio web, con esto, al momento de iniciar Hadoop, ya se podría también entrar en la pestaña *Luminus* de la pantalla de inicio de Hadoop y con esto comenzar a ejecutar las tareas propias de *Luminus* permitiendo de esta forma cargar archivos y ejecutar algoritmos.

Sin embargo, se cambio la forma de Presentar los resultados a una API (Interfaz de Programación de Aplicaciones) por lo que, la forma en la que se incorporen los prototipos 3 , 4 y 5 dependerá de donde se desee que estos sean implementados sobre algún programa Java del usuario final, por lo que, además, no siempre estarían en el mismo equipo que la computadora maestro en el sistema distribuido, con lo que se puede afirmar que, esta administración sería independiente.

E incluso bajo esta nueva definición existe la posibilidad de visualizar la totalidad del trabajo como 2 productos separados pero complementarios. Es decir, se puede requerir al instalador, únicamente para iniciar y configurar una red distribuida con las características para soportar Big Data; y por otro lado o en adición requerir, las configuraciones que tiene la API para soportar los algoritmos sobre la instalación previamente configurada que vendría a ser complementaria pero no forzosa.

Con ello satisfacer necesidades para diferentes grupos de usuarios considerando a los que no desea consumir los algoritmos de minería de datos, o bien, ser complementario, en caso de que el usuario tenga interés en ambos componentes. Por lo que, debido a lo anterior el instalador se elaboro, termino y perfecciono para contemplar la instalación de los prototipos **Evaluación y definición de requerimientos de la red distribuida** y **Integración de requerimientos de la red distribuida**.

En el diagrama de flujo que se muestra en la figura 5.1 se puede apreciar el diseño del instalador.

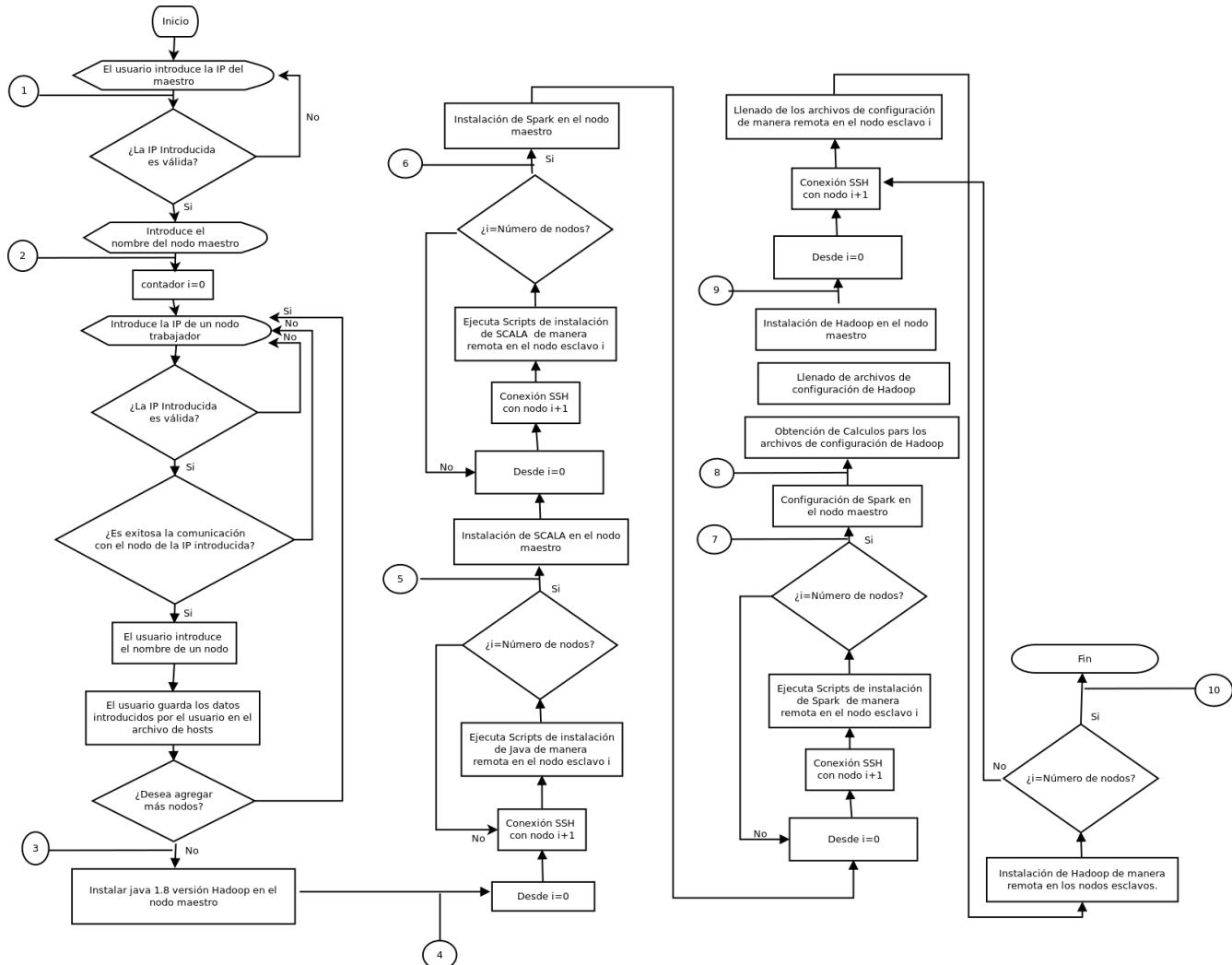


Figura 5.1: Diagrama de flujo del instalador.

El listado de pasos del instalador tal como se puede observar en el diagrama de flujo, si se compará contra el indice del Manual de Instalación de *Luminus*, se podría confirmar que estos en general siguen el mismo proceso y secuencia. Si además se evalúá su flujo paso a paso, sería posible ver que el *Instalador* sigue el mismo flujo que el *Manual de Instalación de Luminus* en cada uno de los pasos, exceptuando que, para algunos de ellos es necesario realizar algunas adecuaciones y consideraciones para que puedan ser ejecutadas desde código y no directamente por el usuario.

Una vez establecido que el listado de pasos es el mismo, faltaría conocer cuales son las consideraciones adicionales que se tienen que tomar en cuenta el instalador para poder operar en relación a hacerlo de manera manual las cuales se encuentran listadas en la sección 5.3.

En Resumen, ambos medios hacen lo mismo, ejecutando los mismos pasos solo que uno de ellos lo realiza de manera automática y el otro requiere intervención del usuario en cada uno de sus pasos. La principal diferencia entre ambos es que el usuario en este instalador no puede manejar las excepciones de manera directa y se busca que el instalador pueda arrojar el menor numero de errores posible y considerar la mayor cantidad de casos,de manera que a medida de nuestras posibilidades, estos puedan ser manejados y resueltos desde el propio código del instalador.

El *Manual de instalación de Luminus* tiene documentados cada uno de los pasos necesarios para el funcionamiento describiendo:

- Que hace cada uno de los pasos en la instalación
- Como lo hace
- Como se configura
- Que significa cada uno de los parámetros de entrada
- Entre otros

Es posible que en caso de existir, dudas, interés o algún problema con el *Instalador de Luminus* se recurra a el *Manual de instalación de Luminus* a consultar la documentación de los pasos que hace.

Por otro lado, también es de utilidad en caso de desear conocer el detalle de cuales son los pasos para instalar alguna de las paqueterías que son instaladas por el instalador, o bien, lo que se requiere para replicar su instalación en los nodos de datos/replica, toda esta información se encuentra en el *Manual de instalación de Luminus*. por lo que puede ser utilizado como referencia.

Una de las consideraciones importantes que este instalador toma en cuenta es que todos los pasos puedan ser ejecutados desde el nodo maestro, para que con esto, sea mas sencillo para el usuario final, en lugar de realizar cada uno de los pasos en los diferentes equipos de computo involucrados esto lo hace haciendo uso de un programa llamado SSH el cual le permite realizar todas las conexiones pertinentes entre los diferentes equipos de computo involucrados.

Esto con el objetivo de que el nodo maestro pueda comunicarse remotamente a los nodos de datos/replica y realizar las instalaciones y configuraciones pertinentes.

Las entradas y salidas del diagrama de flujo con respecto a la intervención del usuario en el manual de instalación actual de Luminus, disminuyen demasiado lo cual será detallado y revisado a profundidad en la sección 5.4.

Explicación del funcionamiento del instalador

A continuación se da una descripción a gran escala de como el instalador lleva a cabo su trabajo en cada uno de los pasos, considerando de manera general, un paso como los que vienen indicados en el diagrama,con un circulo el cual contiene el numero de paso.

1. Este paso consiste en comenzar el instalador y llevar a cabo la primera tarea para que inicie su ejecución la cual seria, solicitar la dirección IP de la computadora que tendrá el papel de nodo maestro.
Como tal no es un paso de ejecución o completa ninguna tarea en particular, sin embargo, sirve para marcar el inicio de la ejecución del instalador, e identificar en caso de fallo que ya se había intentado realizar la instalación anteriormente.
2. Este paso, permite introducir los datos que identifican al nodo maestro, por lo que, una vez completado este paso se tiene conocimiento de la IP del nodo maestro así como del nombre que se le quiere dar como identificador a este nodo.

La información de la IP introducida pasa a través de una expresión regular, la cual, permite validar que efectivamente tiene las características de una dirección IP, por ejemplo: 4 octetos, caracteres numéricos, octetos separados por punto, valor de un octeto no mayor a 255, etc. en caso de que los datos recibidos no sean aprobados en esta validación se solicita al usuario que la ingrese nuevamente.

Si la IP introducida es correcta, se valida que al realizar un ping exista una respuesta, esto para verificar que efectivamente se trate de una IP accesible y asignada a algún equipo de computo dentro de la red.

Una vez hechas ambas validaciones se solicita un nombre de nodo maestro, para identificar al nodo que se acaba de agregar, a continuación se permite al usuario continuar con el instalador y lo posiciona en el punto 2.

3. El siguiente paso, consiste en realizar una operación muy parecida a la que se lleva a cabo en el punto 2, pero en lugar de ser para el nodo maestro se ejecuta para los nodos de datos/replica por lo que, al poder existir más de un nodo de datos/replica esta sección es capaz de repetirse hasta que el usuario haya terminado de ingresar todos los nodos de datos/replica que deseé.

La información de la IP introducida para cada uno de los nodos de datos/replica pasa a través de una expresión regular, la cual, permite validar que efectivamente tiene las características de una dirección IP, por ejemplo: 4 octetos, caracteres numéricos, octetos separados por punto, valor de un octeto no mayor a 255, etc. en caso de que los datos recibidos no sean aprobados en esta validación se solicita al usuario que la ingrese nuevamente.

Si la IP introducida es correcta, se valida que al realizar un Ping exista una respuesta, esto para verificar que efectivamente se trate de una IP accesible y asignada a algún equipo de computo dentro de la red.

Una vez hechas ambas validaciones se permite al usuario indicar el identificador que desea dar a este nodo de datos/replica, y posterior a ello tiene la opción de seleccionar si quiere ingresar un nuevo nodo de datos/replica. En caso de indicar que desea hacerlo entonces esta sección se repetirá para el nuevo nodo, cuando el usuario ya no quiera agregar mas nodos de datos/replica se le permitirá continuar con el instalador y lo posiciona en el punto 3.

4. Cuando ya se tienen todas las configuraciones necesarias para iniciar la instalación de los paquetes necesarios, el primero en instalarse es JAVA instalándolo de primera instancia en el nodo maestro.

Debido a que muchos usuarios tienen otra instalación de JAVA en sus equipos de computo requerida para otras aplicaciones instalar la versión que se necesita para el instalador, tiene que hacerse de forma independiente, creando una nueva carpeta para su instalación.

Por lo que la versión de JAVA requerida, no sería la principal del equipo y tampoco empataría con otras versiones disponibles esto quiere decir que no se intentaría actualizar las versiones que se tengan en la maquina.

Consultando otras herramientas que también ofrecen soluciones de Big Data como *Cloudera*, la solución mencionada anteriormente es la solución que se implementa para esta problemática. por lo que, tomando como referencia el análisis hecho y el conocimiento de que se trata de una posibilidad ya probada por otros sistemas se recurrió a hacerlo de esta misma forma.

Cuando la versión de JAVA se encuentre correctamente instalada en el nodo maestro, se coloca la instalación en el punto 4.

5. Posteriormente se procede a instalar de manera distribuida para cada uno de los nodos que se tienen la paquetería de JAVA, utilizando la misma estrategia propuesta para el nodo maestro, haciendo uso de una carpeta adicional para poder diferenciar esta instalación de otras que pudiera tener el equipo de computo.

Esta instalación se lleva a cabo comunicándose con el nodo de replica/datos haciendo uso de SSH y con esto, hace el envío de lo requerido para ejecutar la instalación y posteriormente se envían las instrucciones en forma de pasos para la terminal para que esta los ejecute y al finalizar JAVA quede instalado, desde cada uno de los nodos de datos/replica indicados en el paso 2 cuando son invocados para instalación.

Una vez que esta instalación se concluye de forma satisfactoria se coloca el instalador en el punto 5.

6. Otra tecnología importante a considerar durante la instalación es SCALA el cual es un lenguaje de programación moderno multi-paradigma sobre el que pueden ser implementadas muchas operaciones de minería de datos y algoritmos de análisis de datos de gran volumen.

Esta instalación primero se efectúa en el nodo maestro y posteriormente es replicada haciendo uso de SSH a cada uno de los nodos de datos/replica.

En lo que respecta a esta instalación en particular solo fue necesario pasar cada uno de los pasos descritos en el manual de instalación a el instalador sin tener que hacer cambios en la lógica de funcionamiento, ni considerar

excepciones.

Una vez que esta instalación se concluye de forma satisfactoria se coloca el instalador en el punto 6.

7. Lo siguiente que hace el instalador en este punto es: Instalar Apache Spark tanto en el nodo maestro como en los nodos de datos/replica.

Para llevar a cabo esta instalación Apache Spark requiere que un par de archivos de configuración sean modificados, por lo que el instalador crea archivos "Plantilla" Los cuales se crean escribiendo los datos correspondientes a la instalación que se está efectuando. Es importante que se haga de esta forma, ya que, cada instalación llena estos archivos con información diferente.

La información que sea escrita en los archivos antes mencionados depende directamente de la información que fue introducida en los pasos 1 paso1 y 3 paso2 como identificadores de los nodos ya que, Apache Spark debe ser capaz de identificarlos y para ello se utilizan los identificadores proporcionados.

Los archivos plantilla que se generan así como el archivo de instalación de Apache Spark es replicado hacia los nodos de datos/replica, para que estos puedan concretar sus propias instalaciones, y los archivos plantilla generados reemplazan a el archivo original que se encuentra en el directorio sin ningún tipo de configuración. Una vez que esta instalación se concluye de forma satisfactoria tanto en el nodo maestro como en los nodos de datos, se coloca el instalador en el punto 7.

8. Los Archivos "Plantilla" son generados en el paso 7 y también son enviados a los nodos y reemplazados los archivos de los nodos con estos archivos, sin embargo, en este punto estos archivos no han sido reemplazados en el servidor maestro, por lo que este paso, permite garantizar que los archivos de configuración con configuraciones a asignar sean reemplazados por los archivos "Plantilla" previamente generados.

Por otro lado, también es necesario modificar los archivos de variable de entorno, para que el nodo maestro sea capaz de reconocer a Apache Spark como una variable de entorno y con esto tener acceso a él desde cualquier punto del sistema operativo entre otras ventajas que son ofrecidas por esta configuración.

una vez hecho esto, se tiene todo lo necesario para poder iniciar Apache Spark únicamente haciendo uso del comando *start-all.sh*.

Cuando estas configuraciones hayan sido efectuadas de manera exitosa, se coloca el instalador en el punto 8.

9. La última tecnología considerada por el instalador es Apache Hadoop, esta es una tecnología con el proceso de instalación de instalación mas pesado de todos los anteriormente contemplados.

Como primer punto es importante instalarlo de manera satisfactoria en el nodo maestro, para ello , al igual que en Apache Spark se requiere el manejo de archivos "Plantilla", ya que se hacen bastantes configuraciones.

Las configuraciones requeridas, dependen de la información propia del equipo de computo que alojará la instalación, por lo que es necesario, correr un script que obtiene información del sistema operativo la procesa y obtiene la que es importante para los archivos de configuración.

además se requieren otro tipo de configuraciones, algunas de los identificadores de los host dentro de la red distribuida y otras del funcionamiento, por lo que los archivos plantilla se construyen a partir de conjuntar toda esta información.

una vez que se tienen los archivos plantilla antes mencionados, se reemplaza en el nodo maestro los archivos de configuración por los archivos plantilla.

también se agrega la información de este programa a las variables de entorno para que el mismo pueda ser accedido desde cualquier punto dentro del sistema operativo.

Cuando estas configuraciones en el nodo maestro hayan sido efectuadas de manera exitosa, se coloca el instalador en el punto 9.

10. Lo siguiente que hace el instalador en este punto es: Instalar Apache Hadoop en los nodos de datos/replica.

Para ello es necesario replicar los archivos plantilla generados a cada uno de los nodos de datos/replica dentro de la red distribuida.

Realizar la instalación de la plataforma en cada uno de estos nodos y reemplazar los archivos por defecto con los archivos de "Plantilla" que se tienen para realizar estas configuraciones. Una vez que esta instalación se concluye de forma satisfactoria en los nodos de datos, se coloca el instalador en el punto 10. es necesario ponerlo en este punto, ya que, si alguien desea volver a iniciar el instalador, con esta información el instalador será capaz de saber que ya había terminado de ejecutar todas las tareas y no quedaban tareas restantes, por lo que se regresará la solicitud notificando que esta instalación ya había terminado con éxito anteriormente.

Este es el ultimo paso del instalador y una vez que este termina entonces termina la ejecución del instalador.

Cada uno de los pasos, descritos anteriormente, al ser completado, marca un estatus de progreso en la ejecución del instalador, este permite que , si el instalador o el proceso de instalación presenta alguna falla o problemática que no le permita continuar con su ejecución, será posible volver a iniciar el instalador y que este reconozca el ultimo punto donde estuvo trabajando para continuar a partir de ese punto y re intentar la instalación. Pero sin comenzar desde el principio y conservando los pasos que resultaron exitosos.

5.4. Desarrollo

El código fuente que se escribió en Shell Script contiene los siguientes pasos que se enuncian en el manual de instalación de *Luminus*.

- 1 Instalación de Apache Spark en el nodo maestro
 - 1.1. Instalación de la paquetería de java
 - 1.3. Instalación de Spark
 - 1.3.1. Configuración maestro
2. Instalación de Apache Spark en los nodos de datos
 - 2.1. Instalación de la paquetería de java en los nodos de datos
 - 2.3. Instalación de Spark en los nodos de datos
 - 2.3.1. Configuración
3. Puesta en funcionamiento
 - 3.1. Configuraciones para poner en funcionamiento Apache Spark en la red distribuida
 - 3.2. Puesta en funcionamiento del cluster manejado por el Servidor Apache Spark
4. Instalación de Apache Hadoop en el nodo maestro
 - 4.1. Instalación de Hadoop
 - 4.1.1. Configuración
 - 4.1.2. Archivos de configuración
5. Instalación de Apache Hadoop en los nodos de datos
 - 5.1. Instalación de Hadoop en los nodos de datos
 - 5.1.1. Configuración

Algunos pasos enunciados en el manual de instalación de *Luminus*, no se encuentran detallados en esta parte, por lo que a continuación se procederá a explicar cuales son estos puntos y la razón por la cual no forman parte del instalador. Estos pasos, tendrían que ser resueltos de manera manual por el usuario final durante el proceso de instalación.

- 1 Instalación de Apache Spark en el nodo maestro
 - 1.2. Instalación de la paquetería de SSH
 - 1.2.1. Configuración
 - 1.2.2. Conexión
2. Instalación de Apache Spark en los nodos de datos
 - 2.2 Instalación de la paquetería SSH en los nodos de datos
6. Puesta en funcionamiento
 - 6.1. Puesta en funcionamiento del Cluster manejado por el Servidor Apache Hadoop
 - 6.2. Subir un archivo al HDFS

Instalación de la paquetería de SSH

Para el instalador, es muy importante que tanto el nodo maestro como los nodos de datos/replica tengan instalado y configurado correctamente SSH ya que en caso de no ser así, no se podría establecer la conexión entre los nodos y por lo tanto sería imposible realizar la instalación de manera distribuida a cada uno de los nodos.

Por lo que se requiere que se encuentre correctamente instalado, el servicio se encuentre iniciado al momento y que además se permitan conexiones de tipo root en cada uno de los nodos.

Por otro lado, se requiere la creación de una llave keygen en el servidor SSH para que las contraseñas que se utilicen

Tarea	Número de pasos en el manual de instalación	Número de pasos en el instalador
Instalación de la paquetería java en el Nodo Maestro	2 pasos	—
Instalación de la paquetería SSH en el Nodo Maestro	9 pasos	9 pasos
Instalación de Spark en el Nodo Maestro	7 pasos	—
Instalación de la paquetería java en los nodos de datos	2 pasos	—
Instalación de la paquetería SSH en los nodos de datos	4 pasos	4 pasos
Instalación de Spark en los nodos de datos	7 pasos	—
Instalación de Apache Hadoop en el nodo maestro	20 pasos	—
Instalación de Apache Hadoop en los nodos de datos	5 pasos	—
Puesta en funcionamiento (Pasos necesarios)	2 pasos	2 pasos
Pasos adicionales del instalador	—	5 pasos
TOTAL	58 pasos	20 pasos

para establecer las conexiones sean seguras y se encuentren cifradas esto con el fin de proteger los datos internos de la empresa y la información que viaja a través de la red, permitiendo cambiar entre los algoritmos de cifrado a utilizar y la longitud con la que las llaves son generadas.

como segundo punto, simplifica el numero de pasos del instalador, ya que de esta manera, no se necesita introducir las contraseñas de root para establecer las conexiones al momento de llevar a cabo el proceso de instalación.

Puesta en funcionamiento

Una vez que se terminan de hacer todas las configuraciones básicas consideradas en el proyecto y se tienen todos los programas necesarios para el mismo, el instalador termina, y con esto es suficiente para que la plataforma de Big Data, este en operación, sin embargo, puede presentarse el caso en que un usuario final requiera realizar mas configuraciones específicas.

En este caso, el usuario, deberá realizar estas configuraciones manualmente antes de poner en funcionamiento, la totalidad del sistema. por lo que, las configuraciones de esta sección tendrían que ser hechas por el usuario, para considerar estas excepciones.

Además de que solo se trata únicamente de 2 pasos los cuales además solo pueden ser hechos en una única ocasión luego de la instalación, y al tratarse de algo tan delicado, por eso lo dejamos como un paso independiente que se considera no es apropiado automatizar.

Consideraciones adicionales

A continuación se muestra una tabla, con el listado del número de pasos que implicaba hacer la instalación de estas tecnologías, con respecto al número de pasos que implica hacerlo mediante el instalador.

Como se puede notar, el número de pasos necesarios disminuyo considerablemente y esto debido a que algunas de las secciones necesarias de llevar a cabo en el manual de instalación, con el instalador ya no son necesarias.

Por otro lado, muchos de los pasos enlistados para la configuración de SSH, puede que algún usuario ya los tenga realizados de instalaciones o configuraciones previas, con lo que en este caso solo sería necesario llevar a cabo **7 pasos** para tener un ambiente de Big Data, funcionando en su totalidad.

Pero, incluso no existiendo este caso ideal, únicamente seria necesario ejecutar el 34 % de los pasos originales.

cuales nos permitirán instalar Apache Spark para permitir la conexión entre los nodos de datos/replica y el maestro haciendo uso de una red de internet local en la que se encuentren conectados todos los nodos.

Además de contener todas las configuraciones necesarias para tal objetivo.

El código fuente actual de este desarrollo se puede consultar en el anexo **8.2**.

Con esto, se logro que el usuario final realice menos pasos de los que tendrían que llevarse a cabo si se siguiera el manual de instalación directamente.

Debido a que las versiones de los software a utilizar pueden estar en constante cambio, y que en el momento de ocurrir una actualización de versión en alguno de los software requeridos por el instalador podrían ocurrir fallas en el instalador

al no soportar los cambios realizados.

Por lo que, se propone utilizar versiones estáticas de el software requerido por el instalador, para que de esta manera se pueda garantizar, actualizar las versiones del instalador en paralelo con las actualizaciones que se realicen sobre la paquetería necesaria para su operación.

De esta forma, se puede garantizar, que el instalador sea capaz de soportar la versión que por el mismo descargue, y no por una actualización del repositorio origen este deje de funcionar de manera correcta o incluso, no existiría la necesidad de buscar entre diferentes servidores para localizar donde se encuentre disponible el archivo que se esta buscando, ya que la información que cada uno de estos servidores independientes tiene disponible no es gestionada por *Luminus*.

Al momento de descargar el instalador desde el servidor que lo aloje, se descargarán también los paquetes requeridos de las tecnologías a utilizar de esta versión.

Dichos paquetes se irían actualizando en el servidor de versiones de *Luminus* para que, con ello, para cada nueva versión que se genere de *Luminus* se actualice también la paquetería necesaria.

Por ejemplo, para la primera versión de este instalador se utilizarían las siguientes versiones de los paquetes a instalar:

- Java Open JDK 1.8
- Scala 2.6.11
- Spark 2.7
- Hadoop 3.1.1

5.5. Pruebas

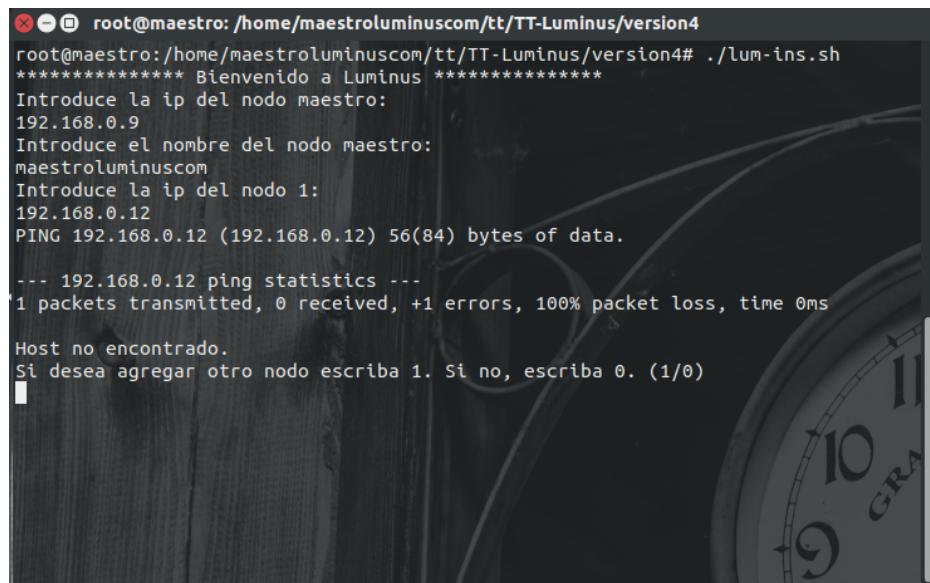
Se presenta a continuación un ejemplo de instalación con el proceso actual a seguir por parte del instalador, así como una descripción de los pasos que este ejecuta.

El instalador será ejecutado solamente desde la máquina que ocupe el rol de nodo maestro de la red distribuida haciendo uso de su terminal con privilegios de super usuario y ejecutando el siguiente comando.

```
./lum-ins.sh
```

Éste preguntará al usuario por la IP del nodo maestro. Luego preguntará por las IPs de los nodos que conformarán la red distribuida. Cada que el usuario introduzca una IP, el prototipo hará un ping para comprobar si hay conexión con el nodo cuya IP se desea agregar a la red distribuida. Si hay respuesta por parte del nodo en cuestión, se procede a almacenar ese dato en un archivo y se le pregunta al usuario si desea agregar otro nodo. Si su respuesta es afirmativa, este proceso se repite.

El instalador, es capaz de detectar cuando la IP introducida no corresponde a un Host que se encuentre dentro de la red local y notifica al usuario experto de esto, como se muestra en la figura 5.5, sin embargo, la salida cuando se puede encontrar el host es diferente, como se puede observar en la figura 5.5.



```

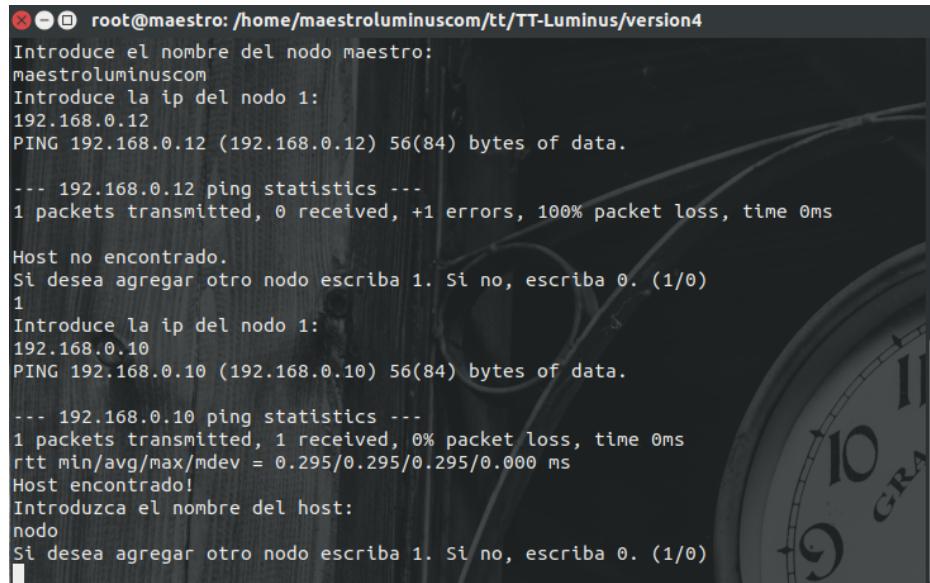
root@maestro: /home/maestroluminuscom/tt/TT-Luminus/version4
root@maestro:/home/maestroluminuscom/tt/TT-Luminus/version4# ./lum-ins.sh
***** Bienvenido a Luminus *****
Introduce la ip del nodo maestro:
192.168.0.9
Introduce el nombre del nodo maestro:
maestroluminuscom
Introduce la ip del nodo 1:
192.168.0.12
PING 192.168.0.12 (192.168.0.12) 56(84) bytes of data.

--- 192.168.0.12 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

Host no encontrado.
Si desea agregar otro nodo escriba 1. Si no, escriba 0. (1/0)

```

Figura 5.2: El usuario introdujo la IP de un host no encontrado.



```

root@maestro: /home/maestroluminuscom/tt/TT-Luminus/version4
Introduce el nombre del nodo maestro:
maestroluminuscom
Introduce la ip del nodo 1:
192.168.0.12
PING 192.168.0.12 (192.168.0.12) 56(84) bytes of data.

--- 192.168.0.12 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

Host no encontrado.
Si desea agregar otro nodo escriba 1. Si no, escriba 0. (1/0)
1
Introduce la ip del nodo 1:
192.168.0.10
PING 192.168.0.10 (192.168.0.10) 56(84) bytes of data.

--- 192.168.0.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.295/0.295/0.295/0.000 ms
Host encontrado!
Introduzca el nombre del host:
nodo
Si desea agregar otro nodo escriba 1. Si no, escriba 0. (1/0)

```

Figura 5.3: El usuario introdujo la IP de un host encontrado.

Posteriormente, cuando ya no se deseen agregar más nodos, el instalador ejecutará varios scripts de instalación en el nodo maestro, desde donde fue ejecutado proceso que puede ser observado en la imagen 5.5.

```
root@maestro:/home/maestroluminuscom/tt/TT-Luminus/version4
Resolving d3kbcqa49mib13.cloudfront.net (d3kbcqa49mib13.cloudfront.net)... 143.2
04.165.85, 143.204.165.100, 143.204.165.181, ...
Connecting to d3kbcqa49mib13.cloudfront.net (d3kbcqa49mib13.cloudfront.net)|143.
204.165.85|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 195636829 (187M) [application/x-tar].
Saving to: 'spark-2.1.0-bin-hadoop2.7.tgz'

58   0K ..... 0% 778K 4m6s
59   50K ..... 0% 897K 3m49s
60 100K kontador=0 ..... 0% 3.20M 2m52s
61 150K echo "" >> /opt/spark/conf/slaves ..... 0% 1.16M 2m49s
62 200K write read line ..... 0% 7.35M 2m20s
63 250K ..... 0% 4.43M 2m4s
64 300K ..... 0% 1.74M 2m2s
65 350K ..... 0% 1.76M 2m0s
66 400K ..... 0% 6.68M 1m49s
67 450K ..... 0% 6.28M 1m41s
68 500K ..... 0% 1.34M 1m45s
69 550K ..... 0% 3.75M 1m40s
70 600K ..... 0% 3.48M 96s
71 650K ..... 0% 875K 1m45s
72 700K ..... 0% 3.73M 1m41s
73 750K ..... 0% 3.04M 99s
```

Figura 5.4: Comienza el proceso de instalación.

Estos scripts instalarán las tecnologías necesarias para que *Luminus* pueda funcionar de manera adecuada. Estas tecnologías son las siguientes:

1. SSH.
2. Java.
3. Scala.
4. Spark.

Después se procede a establecer conexiones SSH con los nodos de la red. Mediante SSH se ejecutan los mismos scripts que se ejecutaron en el nodo maestro pero de manera remota, es decir, sin la necesidad de tener almacenado el script en el nodo donde se va a iniciar la puesta en marcha del ambiente de análisis de datos.

Posteriormente se configura Spark en el nodo maestro, es decir, se asignan roles a los nodos de la red que introduce el usuario al sistema en un principio.

Finalmente se realiza la instalación y configuración de Hadoop en el nodo maestro. Y por último en los nodos de datos/replica.

Una vez que este proceso de instalación finaliza se tiene instalado los prototipos [Evaluación y definición de requerimientos de la red distribuida](#) y [Integración de requerimientos de la red distribuida](#) los cuales tendrán todas las configuraciones necesarias para que luminus pueda funcionar correctamente.

CAPÍTULO 6

Conclusiones

6.1. Conclusiones

Puede decirse con toda certeza que se concretaron las actividades y entregables como se planificó en el cronograma para Trabajo Terminal 1. Se obtuvieron los resultados esperados, los cuales fueron validados con las pruebas realizadas, por lo que los prototipos trabajan adecuadamente. Durante la realización de este primer entregable, pudimos identificar la necesidad de agregar a nuestros productos esperados un instalador que simplifique la preparación del ambiente de análisis de datos *Luminus* al usuario experto. Actualmente ya se está trabajando en la construcción de ese instalador, así como en los demás prototipos que sí fueron planificados desde un principio.

CAPÍTULO 7

Trabajo a futuro

7.1. Trabajo a futuro

Pequeñas mejoras e implementaciones que se pueden hacer sobre lo ya propuesto

En este trabajo a pesar de que se intentó cubrir la mayor cantidad de puntos posibles quedaron abiertos otros puntos a desarrollar. Algunos de los puntos que se sugieren para trabajar posteriormente se listan a continuación:

- Hacer un estudio de la cantidad de nodos mas apropiada para un cluster distribuido, antes de que este deje de ser efectivo para entonces recomendar no exceder este numero de equipos.
- Llevar a cabo un análisis completo de cuales son las características mas apropiadas y que impliquen el menor costo posible en la adquisición y mantenimiento de las computadoras que trabajaran en la red distribuida. Con ello ofrecer una propuesta a las empresas, para que consigan equipos de computo con estas características y con ello obtener el mejor rendimiento optimizando su inversión
- Hacer una implementación diferente del instalador de *luminus* haciendo uso incluso de otra tecnología que permita hacer validaciones de errores de forma completa y contemplando una mayor cantidad de casos para que este pueda ser mas efectivo.
Podría tratarse de un instalador con interfaz gráfica el cual podría resultar mas atractivo para un conjunto de usuarios.
- Realizar el desarrollo y adaptación a Luminus de otros algoritmos de minería de datos ya sea de arboles de decisión o de reglas de asociación o bien crear un nuevo conjunto de algoritmos que sean soportados por la plataforma, además de los 2 ya establecidos.
Para que el análisis que se realice a los datos pueda llegar a ser mas completo y existan más opciones para el usuario experto de algoritmos a aplicar a su caso de estudio.
- Graficación de los resultados de salida de los algoritmos para que se puedan ver las salidas de forma mas clara y visual para el usuario final.
El algoritmo KNN podría dibujar las distancias espaciales con respecto a los vecinos cercanos encontrados.
El algoritmo ID3 podría dibujar el árbol de salida gráfico con sus ramas y hojas.
- Realizar una adaptación al algoritmo KNN para que sea capaz de soportar datos ponderados por el usuario. Es decir, para datos que no son nominales pero que sin embargo no son muy cambiantes y tienen un numero finito de entradas diferentes, estas puedan ser definidas como una equivalencia antes de comenzar la ejecución del algoritmo y con ello considerar estos campos dentro de la ejecución del algoritmo.
Por ejemplo, para un dato que habla de las etapas de vida de una persona:

y este únicamente almacena los siguientes valores: Bebe,Infante,Niño,Adolescente,Adulto Joven,Adulto Maduro, Anciano

estos podrían ser ponderados de la forma en que el usuario elija para cada ejecución para el caso del ejemplo podrían ser agregados valores numéricos como se muestra a continuación.

Bebe 1,Infante 2,Niño 3,Adolescente 4,Adulto Joven 5,Adulto Maduro 6, Anciano 7.

siendo estos valores de ponderación responsabilidad directa de quien los asigne, pero que permiten, tener mas opciones para elegir al momento de seleccionar los atributos que pueden ser utilizados por el algoritmo.

- Realizar los ajustes para que se soporten tipos diferentes de entradas de datos, por ejemplo: JSON, XML , Bases de datos.
ya que al momento únicamente se permite trabajar con archivos CSV, Lo cual puede llegar a ser muy limitado al momento de escoger las fuentes de datos disponibles para alimentar el algoritmo.
- Realizar una implementación que permita obtener automáticamente datos de una fuente externa y los actualice en tiempo real dentro del repositorio de datos. Esto para empresas que poseen datos cambiantes y requieren actualización constante de los mismos.

Cabe señalar que además de estas propuestas existen muchas otras que pueden adaptarse a este proyecto y ampliarlo para diferentes finalidades. Las posibilidades son prácticamente infinitas y dependen directamente de las necesidades específicas de cada usuario o empresa.

Las propuestas que aquí se señalan son ideas que se encontraron durante el desarrollo del proyecto y que se considera pueden contribuir en gran medida al mismo tomando en cuenta los objetivos que se establecieron.

Se tiene además una propuesta general que puede ser aplicada para varios de los puntos anteriormente mencionados y además se trata de una forma diferente de visualizar el proyecto.

Una manera de presentarlo de forma diferente: Sitio web

Como se planteo anteriormente, inicialmente el proyecto estaba pensado para ser desarrollado como un sitio web, sin embargo, durante el desarrollo se optó por la opción de darle un comportamiento de API. esto con el fin de proporcionar una funcionalidad diferente.

Por lo que se puede afirmar que la API no es excluyente de el desarrollo del sitio web. y la implementación mas recomendable dependería de cada usuario final y sus necesidades. Podría trabajarse en la implementación de este sitio web para cubrir una mayor cantidad de usuarios y adaptarse de mejor manera a sus necesidades.

En este sentido, debido a que, ya se había pensado en este desarrollo se tiene una propuesta de como podrían diseñarse las pantallas correspondientes al sitio web por lo menos para el algoritmo KNN además de el mapa de navegación de las mismas. mismas que podrían ser usadas como referencia, o bien, ser utilizadas como tal.

El mapa de navegación es el siguiente:

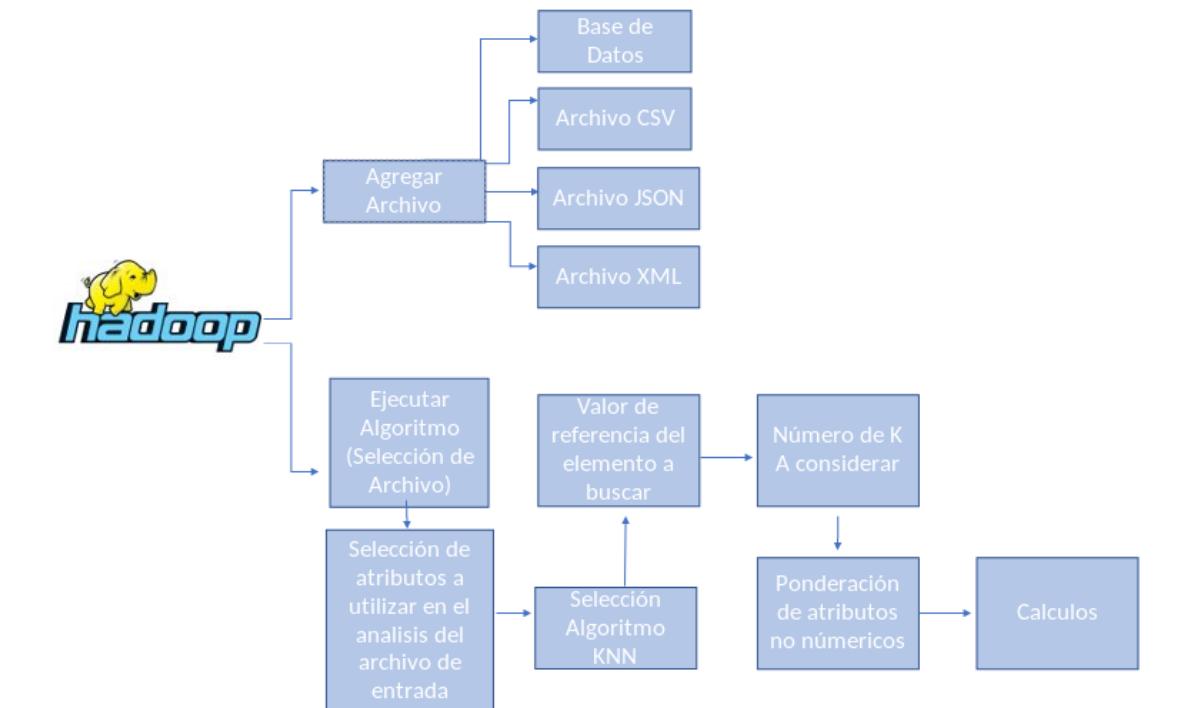


Figura 7.1: Mapa de Navegación

Se procede a explicar cada una de las pantallas que se listan en el mapa de navegación en el orden en que se mandarían a llamar, esto con el fin de que sea claro el proceso de ejecución y funcionamiento de las mismas, además de mencionar para cada una de ellas su objetivo la propuesta de visualización.

El diseño de pantallas propuesto, vendría directamente de las vistas que ya tiene implementadas Hadoop para su funcionamiento, es decir, se agregaría un elemento al menú de opciones de Hadoop. esto para simplificar el entendimiento de las pantallas, y como todas las tareas se realizarían de manera local no sería necesario colgarlas de ningún servicio. En este ejemplo el botón que permitiría llevar a cabo esta tarea sería **Funcionamiento Luminus**, como se muestra en la siguiente figura:



Figura 7.2: Agregar Archivo

El cual, al hacer clic sobre de el, debería redirigir a las pantallas que se muestran a continuación:
Agregar Archivo

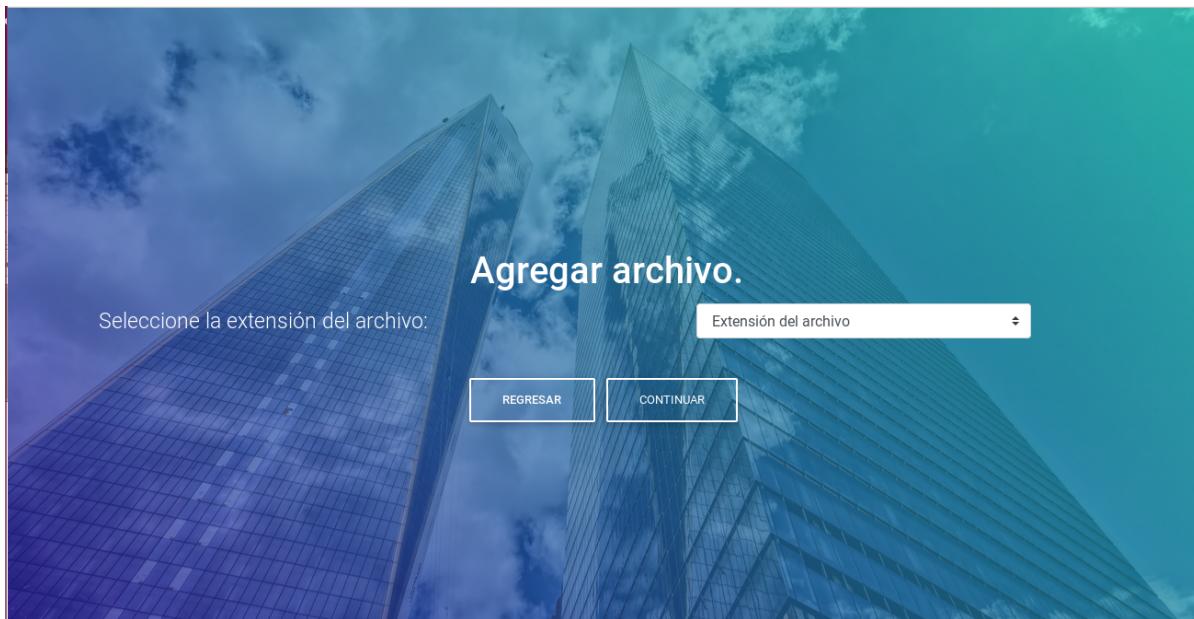


Figura 7.3: Agregar Archivo

Esta pantalla se propone que el usuario seleccione el tipo de archivo que desea subir al HDFS. desde su directorio local, o en su defecto desde una base de datos las configuraciones de conexión a la base de datos serian hechas en pantallas posteriores.

Es necesario que el usuario agregue al menos un archivo de datos al HDFS antes de empezar a aplicar algoritmos ya que en caso de no existir ningún archivo sobre el HDFS, no se tendrían datos sobre los cuales trabajar.

Además de que es importante señalar que, los archivos que el usuario tenga en su directorio local no pueden ser leídos o manipulados por LUMINUS, ni por los algoritmos de programación, por lo que este paso es muy importante.

Con esto, se estaría garantizando que los archivos se encuentren sobre la red distribuida.

[Agregar Archivo CSV](#)



Figura 7.4: Agregar Archivo CSV

En el caso que el archivo que deseé subir el usuario sea de tipo CSV que son los que actualmente son soportados por la plataforma, no se tendría que realizar ningún ajuste o mejora a los algoritmos.

Como parte de la funcionalidad de esta pantalla se propone realizar un análisis detallado a los datos contenidos en el archivo para que, con el análisis que se haga, se le puede decir al usuario que tipo de dato se cree que contiene cada una de las columnas de su archivo, además de permitirle la opción de cambiar el tipo de dato en caso de que el tipo de dato encontrado al ser validado por el usuario este encuentre que es incorrecto. por otro lado, se propone mostrar un ejemplo de un dato contenido dentro de la columna, esto para que si el usuario no recuerda exactamente que contiene una columna, pueda darse una idea y con esto seleccionar si el tipo de dato que se encontró para esa columna efectivamente es el correcto y en caso de no serlo pueda tener los elementos para cambiarlo con toda tranquilidad.

Para los archivos CSV que no contienen encabezado con nombre de los atributos, se mostrara esta columna vacía y se dejará al usuario que indique su nombre.

Una vez terminado el proceso podrá hacer clic en uno de los dos botones listados; ya sea en **Aceptar y Terminar** o bien en **Regresar**. y esto lo determina, el hecho de que desee o no, guardar el archivo encontrado con los datos que se encuentran actualmente en la tabla mostrada en la pantalla.

En caso de que se presione **Aceptar y Terminar** esta pantalla debería validar que los tipos de dato seleccionados por el usuario puedan ser aplicados para todos los elementos que contiene cada uno de los atributos y si no lo es, mostrar uno de los elementos del atributo que no aplica para el tipo de dato introducido por el usuario, para que entonces el usuario final pueda volver a hacer la selección.

En caso de que todos los tipos de dato sean correctos, se debería hacer una carga de este archivo al HDFS, agregando en caso de que no lo tuviera y hayan sido introducido por el usuario los atributos correspondientes.

Los tipos de datos, seleccionados por el usuario, serían utilizados para alimentar los algoritmos soportados por *Luminus* con los datos que cada uno de ellos soporta. y se tendría que llevar un control de estos tipos de datos, como el programador de este sitio web lo prefiera siempre y cuando no los pierda de vista para tomarlos en cuenta al momento de invocar a los programas.

Agregar XML o Agregar JSON

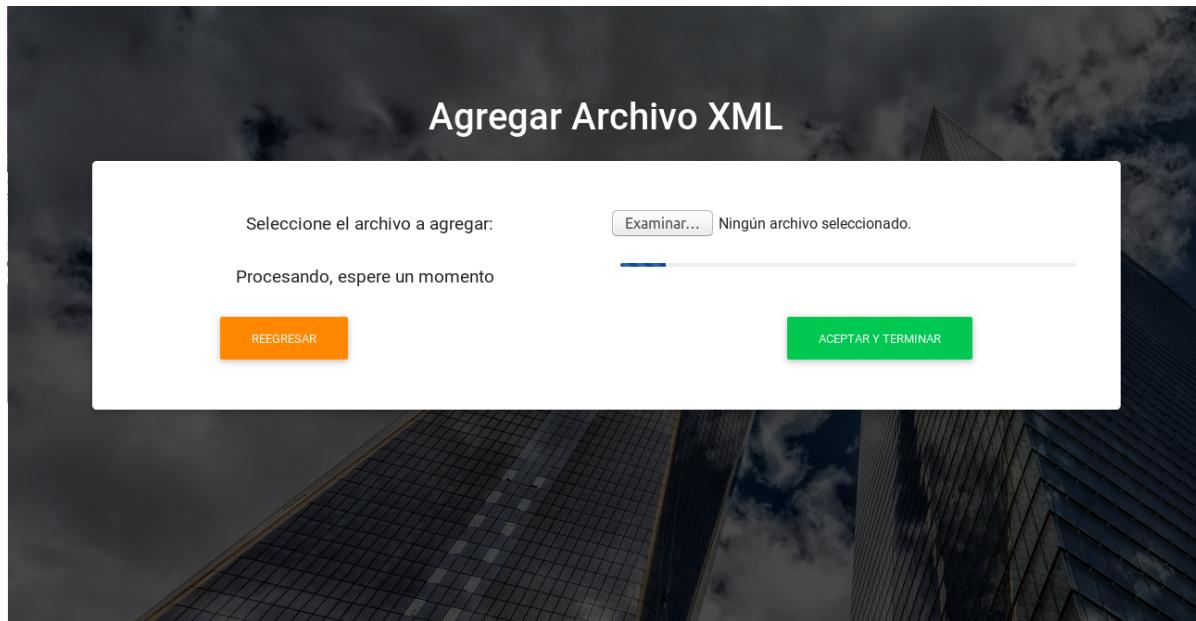


Figura 7.5: Agregar Archivo XML

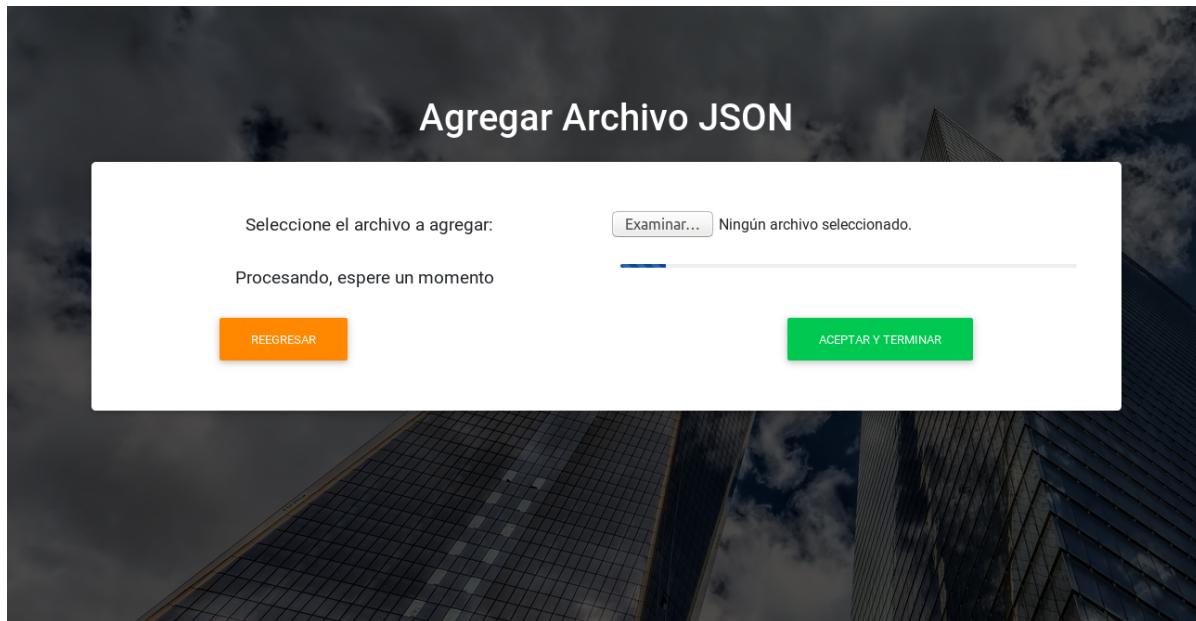


Figura 7.6: Agregar Archivo JSON

Estos tipos de archivo, deben ser cargado desde el directorio local del usuario del sitio web, y no debería requerir ningún tipo de configuración adicional ya que como tal, estos tipos de archivo están diseñados de tan forma que la información que se desea conocer del archivo ya está soportada en la propia estructura del mismo, por lo que no habría que pedir o validar detalles adicionales.

Sin embargo, este tipo de archivos aun no son soportados por los algoritmos implementados en *Luminus*, por lo que, sería necesario realizar alguna de las siguientes opciones:

- Adaptar los algoritmos que actualmente se tienen implementados en *Luminus* para que sean capaces de trabajar

con archivos XML y JSON

- Hacer un tratamiento de los datos para que a pesar de que su origen sea un tipo de archivo distinto (XML, JSON) puedan ser tratados como un CSV y soportados por los algoritmos.

Sin embargo, sin importar la solución que se escoja será necesario realizar la subida de los archivos al directorio del HDFS.

Agregar Base de Datos



Figura 7.7: Agregar Tabla o Vista de una Base de Datos

Algunas veces los usuarios, tendrán sus datos en bases de datos relacionales convencionales. ya que es una manera muy común que utilizan las empresas para el tratamiento de sus datos, en este caso, se puede destinar una tabla o una vista para poder trabajar con los datos que estas nos arrojan.

La vista, en su caso, le permitiría al usuario final, incluir solo algunos campos de una determinada tabla, o bien datos de varias tablas haciendo las uniones entre las tablas que este necesite.

Por lo que una vez que el usuario tuviera lista su vista o bien su tabla, podrá en esta pantalla ingresar las credenciales de su base de datos, las cuales son:

- Dirección IP

- Puerto donde corre la base de datos
- Nombre de usuario
- Clave de este usuario dentro de la base de datos

Bases de datos de diferentes manejadores, pueden establecer una conexión únicamente con estos 4 datos por lo que se considera que estos podrían ser suficientes para testear una conexión.

Antes de realizar el Testeo, será necesario que el usuario indique cual es el nombre de la tabla o vista con la que desea conectarse, después de esto, se puede verificar que la conexión a la base de datos es valida y continuar con el copiado de la tabla o de la vista dentro de el HDFS.

La implementación para base de datos, no es soportada actualmente por los algoritmos de clasificación programados, por lo que se tendría que implementar alguna de las siguientes soluciones:

- Adaptar los algoritmos de *Luminus* para que sean capaces de trabajar con archivos originarios de una base de datos ya sea una tabla o una vista.
- Hacer un tratamiento de los datos para que a pesar de que su origen sea un tipo de archivo distinto (Base de datos) puedan ser tratados como un CSV y soportados por los algoritmos.

En este caso, es necesario considerar, cuantos tipos de bases de datos diferentes pueden ser soportados por los algoritmos debido a que cada base de datos que se maneje tiene diferentes formas de exportar sus datos, y este comportamiento deberá ser considerado al momento de manejarlos.

Ejecutar algoritmo (Selección de Archivo)

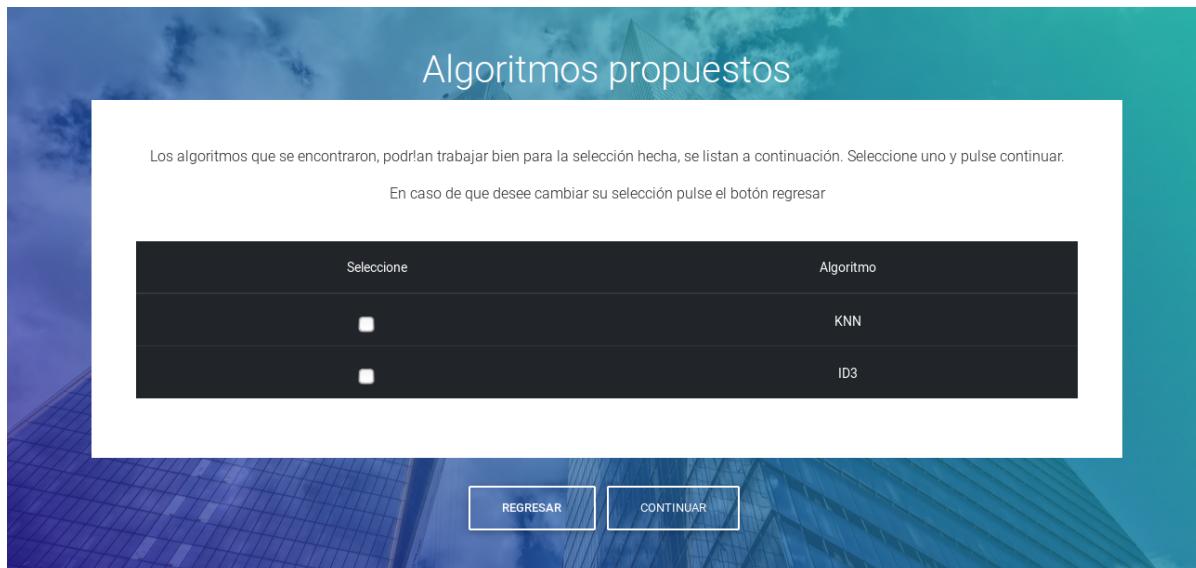


Figura 7.8: Algoritmos Propuestos

En esta pantalla se pretende presentar todos los tipos de algoritmos que pueden ser soportados por la aplicación, por el momento ID3 y KNN, en caso de que, se desarrollen mas algoritmos podrían ser listados desde esta misma pantalla.

Sería necesario que el usuario seleccionará del listado mostrado el algoritmo que desea ejecutar.

Algoritmo KNN

A continuación se muestra un ejemplo de como podría operar el algoritmo KNN en esta implementación. no se tiene una propuesta para el algoritmo ID3 pero la lógica que estos debería ser la misma.

Si se alcanza a visualizar la forma en que se pretende presentar este algoritmo, podría diseñarse una solución para cada uno de los algoritmos que sean implementados.

Ejecutar Algoritmo



Figura 7.9: Algoritmos Propuestos

En este caso, se buscará en el HDFS, el archivo que contiene los datos que se desean aplicar al algoritmo KNN, ya que, el HDFS puede contener mas de un archivo de datos. pero será necesario distinguir cual de ellos sera utilizado para cada ejecución. Una vez que este se seleccione se presionaría **Continuar** .

Selección de atributos a utilizar en el análisis del archivo de entrada



Figura 7.10: Selección de atributos

En esta pantalla, se muestran los atributos contenidos dentro del archivo de configuraciones, para que el usuario seleccione los que desea utilizar como parte del procesamiento.

Se propone restringir a que no sea posible seleccionar mas de 5 atributos, ya que entre mas atributos se incluyan en

el algoritmo mas complejo se vuelve su procesamiento, sin embargo, podría no ser restringido y seguiría operando de manera correcta.

Nuevamente, si el usuario desea cambiar el tipo de dato, debería ser permitido. ya que quizá deseé tratar el mismo dato de diferentes maneras para diferentes algoritmos. Pero sería recomendable mantener los que seleccionó anteriormente, para que solo requiera hacer los cambios para los atributos que tengan modificaciones y el resto de ellos no tenga que volver a introducirlos cada que quiera ejecutar un algoritmo.

Valor de referencia del elemento a buscar



The screenshot shows a mobile application interface titled "Valor de referencia". The background features a blue-toned image of a modern skyscraper. The main content area has a white background with a dark header bar. The header bar contains the title "Valor de referencia" in white text. Below the header, there is a text input field with the placeholder "Escriba el valor de referencia que quiere utilizar para su análisis". Underneath the input field is a table with three columns: "Seleccione", "Tipo de dato", and "Valor". The table has three rows. The first row contains "Edad" in "Seleccione", "Numerico" in "Tipo de dato", and a blank input field in "Valor". The second row contains "Sueldo" in "Seleccione", "Decimal" in "Tipo de dato", and a blank input field in "Valor". At the bottom of the screen are two buttons: "REGRESAR" on the left and "CONTINUAR" on the right.

Figura 7.11: Selección de atributos

Para todos los datos seleccionados en la pantalla anterior se tiene que dar un valor de referencia para el análisis, este servirá para encontrar el número de vecinos más cercanos. el valor que se ingrese debe ser del mismo tipo de dato, seleccionado en la pantalla anterior.

Número de K a considerar



Valor de referencia

Ingrese el número de K con el que desea alimentar el algoritmo, para un número de K más grande el algoritmo puede tomar mas tiempo
 Los valores recomendados estan entre 1 y 11, valores mayores a 11 pierden significado en los resultados.
 Se recomienda usar un número impar

Seleccione

 Valor de K

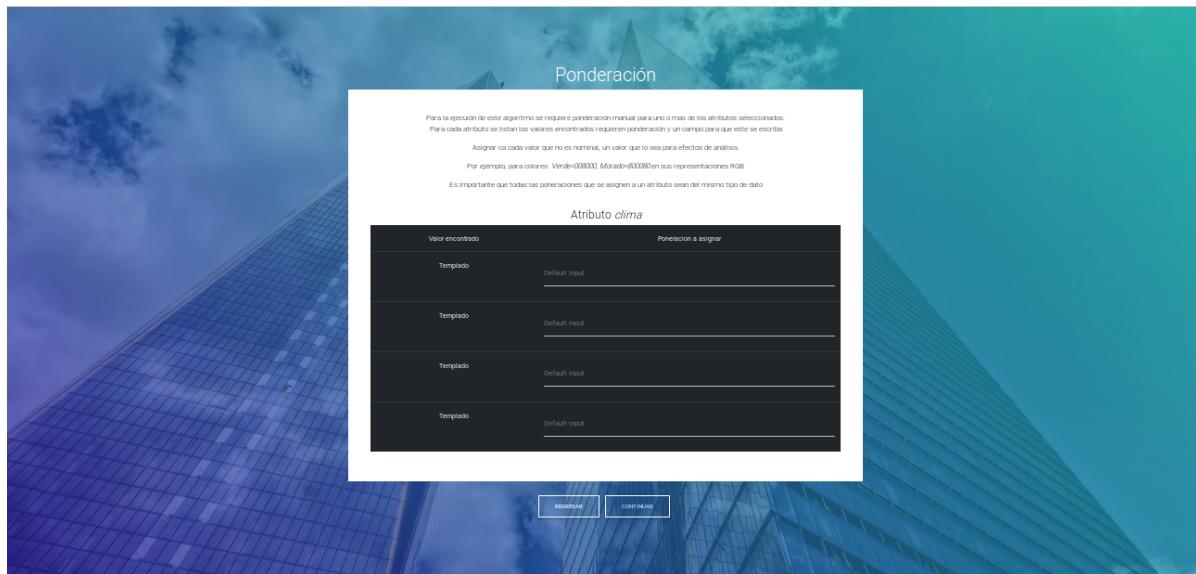
[REGRESAR](#)
[EJECUTAR](#)

Figura 7.12: Valor K

En este caso, para obtener los valores se selecciona un numero de K valido, se recomienda seleccionar este valor entre 0 y 11. para hacer mas simple el análisis. ya que valores de k superiores a 11 la información empieza a perder valor.

esta pantalla únicamente permite seleccionar el valor mas apropiado para esta variable.

Ponderación de atributos no numéricos



Ponderación

Para la ejecución de este algoritmo se requiere ponderación manual para uno o más de los atributos seleccionados. Para cada atributo se instan los valores encontrados requieren ponderación y un campo para que este sea escrito. Agregar co cada valor que no es homónimo, un valor que lo sea para efectos de análisis. Por ejemplo, para colores: Verde#000000, Morado#0000ff en sus representaciones RGB. Es importante que todas las ponderaciones que se asignen a un atributo sean del mismo tipo de dato.

Atributo clima	Ponderación a asignar
Valor encontrado: Templado	Default Input
Templado	Default Input
Templado	Default Input
Templado	Default Input

[REGRESAR](#)
[CONTINUAR](#)

Figura 7.13: Ponderación

Un elemento que se puede agregar a las consideraciones hechas es la ponderación de atributos no numéricos. Se trata de valores, que no pueden ser trabajados de manera inmediata por los algoritmos, por lo que hay que asignarles un valor de manera manual a cada uno de los elementos del atributo. Esto es viable mientras que, el número de elementos dentro de un atributo no haya crecido demasiado, ya que para atributos con muchas opciones esto se volvería muy pesado.

Sería necesario establecer un número máximo de valores diferentes permitidos para que, atributos que tengan mas elementos que ese numero establecido no puedan ser configurados de esa forma.

Todas las ponderaciones que se establezcan, deben de ser del mismo tipo de dato, y debe de configurarse el archivo de tal forma que cuando se pase a los algoritmos pueda leer los datos ponderados para que el algoritmo que se pretende utilizar funcione de manera correcta.

Al inicio de la sección Trabajo a Futuro se detalla un poco mas acerca de que significa hacer las ponderaciones y se detalla un ejemplo en caso de que se requiera mas información al respecto.

Ejecución del algoritmo



Figura 7.14: Ejecución del algoritmo

Cuando se realicen todas las configuraciones necesarias, entonces se procede a ejecutar el algoritmo pasandole las configuraciones establecidas y se devuelve al usuario un archivo de salidas en el formato mas recomendable para el algoritmo que se este ejecutando. con los resultados obtenidos de la ejecución del algoritmo.

Bibliografía

- [1] M. Mittal, V.E. Balas, D.J. Hemanth (2018) *Data Intensive Computing Applications for Big Data*, 2018.
- [2] Manieviesa, P, 2017 *El Big Data en las empresas*, Septiembre 21, 2017 Pymerang.com. Sitio web:<http://www.pymerang.com/direccion-de-negocios/821-el-big-data-en-las-empresas>.
- [3] Joyanes, L (2016) «Big Data, Análisis de grandes volúmenes de datos en organizaciones», *AlfaOmega*,
- [4] PowerData, S/A *Big Data: ¿En qué consiste? Su importancia, desafíos y gobernabilidad*, Septiembre 14, 2017 Pymerang.com. Sitio web: <http://www.powerdata.es/big-data>.
- [5] KNIME, (2017) *open for innovation KNIME*, Octubre 6, 2017, de KNIME Sitio web: <https://www.knime.com/>
- [6] IBM,(2017) *IBM SPSS Modeler*, Octubre 6, 2017,de IBM Sitio web. <https://www.ibm.com/es-es/marketplace/spss-modeler/purchase#product-header-top>
- [7] QlikSense,(2017) *Simple to use. More power than you can imagine*, Septiembre 14,2017, de QlikSense. Sitio web: <http://www.qlik.com/>
- [8] Databricks, (2018) *Databricks Unified Analytics Platform*, Agosto 3, 2018, de Databricks. Sitio web: <https://databricks.com/product/unified-analytics-platform>
- [9] Cloudera, (2018), *About Cloudera*, Enero 23, 2018, de Cloudera. Sitio web: <https://www.cloudera.com/more/about.html>
- [10] Hortonworks, (2018), *Hortonworks Data Platform*, Junio 14, 2018, de Hortonworks. Sitio web: <https://es.hortonworks.com/products/data-platforms/hdp/>
- [11] Databricks, (2018), *Launch cloud-optimized Apache Spark™ clusters in minutes*, Septiembre 20, 2018, de Databricks. Sitio web: <https://databricks.com/try-databricks>
- [12] Cloudera, (2018), *Download Cloudera*, Septiembre 10, 2018, de Cloudera. Sitio web: <https://www.cloudera.com/downloads.html>
- [13] Hortonworks, (2018), *Descargas de las plataformas de datos conectadas de Hortonworks*, Agosto 1, 2018, de Hortonworks. Sitio web: <https://es.hortonworks.com/downloads/>
- [14] Data Science,(2017) *Does a big data virtual machine machine help in analyzing large files*, Septiembre 14, 2017, de Data Science StackExchange. Sitio web <https://datascience.stackexchange.com/questions/14993/does-a-big-data-virtual-machine-machine-help-in-analyzing-large-file>
- [15] Srinivasan, V. (2016) «The Intelligent Enterprise in the Era of Big Data.», Canada: John Wiley & Sons.
- [16] Manchón M.,(2017) *Las empresas y el reto del big data: sólo usan el 1 % de los datos.*, Junio 6, 2017, de EDDeconomíaDigital Sitio web: https://www.economiadigital.es/directivos-y-empresas/empresas-big-data-uno-por-ciento_408504_102.html

- [17] García, S.(2017) *Big Data: realidades, retos y limitaciones*, Julio 16, 2017, de Excelsior Sitio web: <http://www.excelsior.com.mx/opinion/opinion-del-experto-nacional/2017/07/16/1175906>
- [18] The Apache Software Foundation (2018) *Launching Applications Using Docker Containers*, Marzo 16, 2018, de Apache Hadoop. Sitio web: <https://hadoop.apache.org/docs/r3.0.1/hadoop-yarn/hadoop-yarn-site/DockerContainers.html>
- [19] Santos A *Programación Shell*, Septiembre 12, 2018, de freeshell.de. Sitio web: www.freeshell.de/rasoda/programacion/guia-shell.pdf
- [20] Oracle (2017) *The Definition of Big Data*, Marzo 22, 2017, de Oracle. Sitio web: <https://www.oracle.com/big-data/guide/what-is-big-data.html>
- [21] Oracle (2017) *Big Data Use Cases*, Abril 14, 2017, de Oracle. Sitio web: <https://www.oracle.com/big-data/guide/big-data-use-cases.html>
- [22] Sinnexus (2016) *Datamining (Minería de datos)*, Marzo 6, 2016 de Sinnexus. Sitio web: https://www.sinnexus.com/business_intelligence/datamining.aspx
- [23] Microsoft (2017) *Conceptos de minería de datos*, Julio 22, 2017, de Microsoft. Sitio web: <https://docs.microsoft.com/es-es/sql/analysis-services/data-mining/data-mining-concepts?view=sql-server-2017>
- [24] GestiónDeOperaciones (2016) *Árbol de Decisión (Qué es y para qué sirve)*, Marzo 17, 2016 de www.gestiondeoperaciones.net. Sitio web: <https://www.gestiondeoperaciones.net/procesos/arbol-de-decision/>
- [25] LucidChart (2016) *Qué es un diagrama de árbol de decisión*, Diciembre 12, 2016, de LucidChart. Sitio web: <https://www.lucidchart.com/pages/es/qu%C3%A9-es-un-diagrama-de-%C3%A1rbol-de-decisi%C3%B3n>
- [26] Sefik Ilkin Serengil (2017) *A Step by Step ID3 Decision Tree Example*, Noviembre 20, 2017, de Sefik Ilkin Serengil. Sitio web: <https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/>
- [27] Sefik Ilkin Serengil (2018) *A Step By Step C4.5 Decision Tree Example*, Mayo 13, 2018, de Sefik Ilkin Serengil. Sitio web: <https://sefiks.com/2018/05/13/a-step-by-step-c4-5-decision-tree-example/>
- [28] Deepanshu Bhalla (2016) *K NEAREST NEIGHBOR : STEP BY STEP TUTORIAL*, Septiembre 30, 2016, de Deepanshu Bhalla. Sitio web: <https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html>
- [29] Andrea Trevino (2016) *Introduction to K-means Clustering*, Diciembre 6, 2016, de Oracle. Sitio web: <https://www.datascience.com/blog/k-means-clustering>
- [30] IBM, (2018) *Apache MapReduce*, Febrero 8, 2018, de IBM. Sitio web: <https://www.ibm.com/analytics/hadoop/mapreduce>
- [31] Abraham Requena Mesa (2017) *¿Qué es Hadoop?*, Septiembre 28, 2017, de OpenWebinars. Sitio web: <https://openwebinars.net/blog/que-es-hadoop/>
- [32] Abraham Requena Mesa (2018) *Qué es Apache Spark*, Julio 02, 2018 de OpenWebinars. Sitio web: <https://openwebinars.net/blog/que-es-apache-spark/>
- [33] David Loshin (2017) *Explorando distribuciones Hadoop para gestionar big data*, Mayo 30, 2017, de SearchDataCenter. Sitio web: <https://searchdatacenter.techtarget.com/es/cronica/Explorando-distribuciones-Hadoop-para-gestionar-big-data>
- [34] Microsoft (2018) *What is a server cluster?*, Mayo 28, 2018, de Microsoft. Sitio web: [https://technet.microsoft.com/pt-pt/library/cc785197\(v=ws.10\).aspx](https://technet.microsoft.com/pt-pt/library/cc785197(v=ws.10).aspx)
- [35] BTECH (2010) *What is Computing Environment?*, Julio 12, 2010, de BTECH. Sitio web: <http://btechsmartclass.com/CP/computing-environments.htm>
- [36] Devin Soni (2018) *Introduction to Bayesian Networks*, Junio 8, 2018, de Towards Data Science. Sitio web: <https://towardsdatascience.com/introduction-to-bayesian-networks-81031eed94e>

7.2. 1.1 hola

7.2.1. Resumen

lala



7.2.2. Descripción

Caso de Uso:	1.1 hola
Versión:	1.0
Administración de Requerimientos	
Autor:	Adrian Flores Torres
Evaluador:	
Operación:	Registrar
Prioridad:	Alta
Complejidad:	Baja
Volatilidad:	Baja
Madurez:	Alta
Estatus:	En revisión
Fecha del último estatus:	28 Septiembre 2017

7.2.3. Trayectorias del Caso de Uso

Trayectoria principal

- 1  Selecciona la opción "Esquema de Pagos"
- - - - *Fin del caso de uso.*

CAPÍTULO 8

Anexos

8.1. Anexo A: Código JAVA Map-Reduce

```
package PackageDemo;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
    public static void main(String [] args) throws Exception
    {
        Configuration c=new Configuration();
```

```
String [] files=new GenericOptionsParser(c, args).getRemainingArgs();  
  
Path input=new Path(files[0]);  
  
Path output=new Path(files[1]);  
  
Job j=new Job(c, "wordcount");  
  
j.setJarByClass(WordCount.class);  
  
j.setMapperClass(MapForWordCount.class);  
  
j.setReducerClass(ReduceForWordCount.class);  
  
j.setOutputKeyClass(Text.class);  
  
j.setOutputValueClass(IntWritable.class);  
  
FileInputFormat.addInputPath(j, input);  
  
FileOutputFormat.setOutputPath(j, output);  
  
System.exit(j.waitForCompletion(true)?0:1);  
  
}  
  
public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{  
  
    public void map(LongWritable key, Text value, Context con) throws  
IOException, InterruptedException  
  
    {  
  
        String line = value.toString();  
  
        String [] words=line.split(",");  
  
        for(String word: words)  
  
        {  
  
            Text outputKey = new Text(word.toUpperCase().trim());  
  
            IntWritable outputValue = new IntWritable(1);  
  
            con.write(outputKey, outputValue);  
  
        }  
  
    }  
  
}
```

```
public static class ReduceForWordCount extends Reducer<Text,
  IntWritable, Text, IntWritable>

{

  public void reduce(Text word, Iterable<IntWritable> values, Context
  con) throws IOException, InterruptedException

  {

    int sum = 0;

    for(IntWritable value : values)

    {

      sum += value.get();

    }

    con.write(word, new IntWritable(sum));

  }

}

}
```

8.2. Anexo B: Código del instalador

Código de la clase principal luminus-ins.sh

```
#!/bin/bash
# Instalador general de luminus version 4.

# El instalador pide al usuario introducir las ips de los nodos incluida la del master.
# Las ips son almacenadas en un archivo de texto.

# Funcion que valida el formato de las ips mediante el uso de expresiones regulares.
function validar_ip() {
  local valida=0
  local ip=$1
  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    OIFS=$IFS
    IFS='.'
    ip=($ip)
    IFS=$OIFS
    if [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
      && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
    then
      valida=1
      echo $valida
    else
      echo $valida
    fi
  else
    echo $valida
  fi
}
```

```
        fi
    else
        echo $valida
    fi
}

# Lectura de la ip del nodo maestro
function pedir_ip_maestro() {
    # Se escribe un 1 en el log del progreso de instalacion.
    # El 1 corresponde a que solamente se inicio pero no se completo el primer paso.
    echo "1" > luminus-ins.log
    cp /etc/hosts.back /etc/hosts
    confirmacionIPMaestro=0
    while [ $confirmacionIPMaestro -eq 0 ]
    do
        echo "Introduce la ip del nodo maestro:"
        read ipMaestro
        if [ $(validar_ip $ipMaestro) -eq 1 ]
        then
            echo "Introduce el nombre del nodo maestro:"
            read nodoMaestro
            echo $nodoMaestro > nombres.txt
            echo $ipMaestro > ips.txt
            echo "$ipMaestro $nodoMaestro maestro" >> /etc/hosts
            echo "$ipMaestro $nodoMaestro maestro" > hosts.maestro
            cp /etc/hosts /etc/hosts.back
            cp ~/.bashrc ~/bashrc.back
            ssh-keygen -f "/root/.ssh/known_hosts" -R $ipMaestro
            confirmacionIPMaestro=1
        else
            echo "Introduzca una IP valida."
        fi
    done
    # Se escribe un 2 en el log de progreso de instalacion.
    # El 2 corresponde a que se introdujo de manera satisfactoria la IP del nodo maestro
    echo "2" > luminus-ins.log
    pedir_ips_esclavos
}

# Lectura de las ips de los nodos esclavos.
# Escritura de las ips en el archivo hosts del nodo maestro.
# Adicion de la ip del nodo maestro al archivo hosts de los nodos esclavos.
function pedir_ips_esclavos() {
    confirmacion=1
    confirmacionIP=0
    contador=1
    while [ $confirmacion -eq 1 ]
    do
        while [ $confirmacionIP -eq 0 ]
        do
            echo "Introduce la ip del nodo $contador:"
            read ipNodo
            if [ $(validar_ip $ipNodo) -eq 1 ]
            then
```

```

        ping -c1 -qq $ipNodo
        if [ $? -ne 0 ]
        then
                echo "Host no encontrado."
        else
                echo "Host encontrado!"
                echo "Introduzca el nombre del host:"
                read nodo
                echo "$ipNodo $nodo $nodo" >> /etc/hosts
                ssh-copy-id -i ~/.ssh/id_rsa.pub root@$ipNodo
                ssh root@$ipNodo "cp /etc/hosts.back /etc/hosts"
                scp hosts.maestro root@$ipNodo:/etc/
                ssh root@$ipNodo "cat /etc/hosts.maestro >> /etc/hosts"
                ssh root@$ipNodo "rm /etc/hosts.maestro"
                ssh root@$ipNodo "cp /etc/hosts /etc/hosts.back"
                ssh root@$ipNodo "cp ~/bashrc ~/bashrc.back"
                ssh-keygen -f "/root/.ssh/known_hosts" -R $ipNodo
                echo $nodo >> nombres.txt
                echo $ipNodo >> ips.txt
                contador=$((contador + 1))
                confirmacionIP=1
        fi
    else
        echo "Introduzca una IP valida."
    fi
done
echo "Si desea agregar otro nodo escriba 1. Si no, escriba 0. (1/0)"
read confirmacion
done
# Numero de replicacion para el hdfs segun el numero de nodos esclavos
cp config-files/hdfs-site-config.back.xml config-files/hdfs-site-config.xml
numNodos=$((contador - 1))
echo $numNodos >> config-files/hdfs-site-config.xml
echo "</value> </property> </configuration>" >> config-files/hdfs-site-config.xml
# Se escribe un 3 en el log de progreso de instalacion.
# El 3 corresponde a que se introdujeron de manera satisfactoria las IPs de los esclavos
echo "3" > luminus-ins.log
instalar_java_maestro
}

# Instalacion de java 1.8 especifico para hadoop
function instalar_java_maestro() {
    rm -rf /usr/lib/jvm/java-8-oracle-hadoop
    mkdir /usr/lib/jvm
    tar -xvf jdk-8u201-linux-x64.tar.gz
    mv jdk1.8.0_201/ /usr/lib/jvm/java-8-oracle-hadoop
    update-alternatives --install /usr/bin/java java /usr/lib/jvm/java-8-oracle-hadoop/ java
    # Se escribe un 4 en el log de progreso de instalacion.
    # El 4 corresponde a la instalacion de java de manera satisfactoria en el nodo maestro
    echo "4" > luminus-ins.log
    instalar_java_esclavos
}

function instalar_java_esclavos() {

```

```
contador=1
while read line
do
    if [ $contador -ne 1 ]
    then
        scp jdk-8u201-linux-x64.tar.gz root@$line:/home/
        ssh -n root@$line "rm -rf /usr/lib/jvm/java-8-oracle-hadoop"
        ssh -n root@$line "mkdir /usr/lib/jvm/"
        ssh -n root@$line "tar -xvf /home/jdk-8u201-linux-x64.tar.gz"
        ssh -n root@$line "mv jdk1.8.0_201/ /usr/lib/jvm/java-8-oracle-hadoop"
        ssh -n root@$line "update-alternatives --install /usr/bin/java java
        fi
        contador=$((contador + 1))
    done < ips.txt
    cp ~/.bashrc.backup ~/.bashrc

# Se escribe un 5 en el log de progreso de instalacion.
# El 5 corresponde a la instalacion de java de manera satisfactoria en los esclavos
echo "5" > luminus-ins.log
#instalar_scala
}

#Instalacion de Scala en todos los nodos.
function instalar_scala() {
    ./scala-ins.sh
    contador=1
    while read line
    do
        if [ $contador -ne 1 ]
        then
            echo $line
            ssh -qq root@$line < scala-ins.sh
        fi
        contador=$((contador + 1))
    done < ips.txt

# Se escribe un 6 en el log de progreso de instalacion.
# El 6 corresponde a la instalacion de scala de manera satisfactoria en el maestro
echo "6" > luminus-ins.log
instalar_spark
}

# Instalacion de spark en todos los nodos.
function instalar_spark() {
    ./spark-ins.sh
    contador=1
    while read line
    do
        if [ $contador -ne 1 ]
        then
            echo $line
            ssh -qq root@$line < spark-ins.sh
        fi
        contador=$((contador + 1))
```

```
done < ips.txt

# Se escribe un 7 en el log de progreso de instalacion.
# El 7 corresponde a la instalacion de Spark de manera satisfactoria en el maestro
echo "7" > luminus-ins.log
configurar_spark_maestro
}

# Configuracion de Spark en el nodo maestro.
function configurar_spark_maestro() {
    # Copia de la plantilla de esclavos de Spark.
    cp /opt/spark/conf/slaves.template /opt/spark/conf/slaves

    # Se agregan los esclavos y el maestro a la lista de nodos de la red.
    cat ips.txt > /opt/spark/conf/slaves

    # Se copia la plantilla de spark-env
    cp /opt/spark/conf/spark-env.sh.template /opt/spark/conf/spark-env.sh

    # Se agrega el maestro al archivo spark-env.sh
    echo "export SPARK_MASTER_HOST=$ipMaestro" >> /opt/spark/conf/spark-env.sh

    # Se escribe un 8 en el log de progreso de instalacion.
    # El 8 corresponde a la configuracion de Spark de manera satisfactoria en el maestro
    echo "8" > luminus-ins.log
    instalar_hadoop_maestro
}

# Instalacion de Hadoop.
function instalar_hadoop_maestro() {
    # wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-3.1.1/hadoop-3.1.1.tar.gz
    rm -rf /opt/hadoop
    mkdir /opt/hadoop
    tar -xzvf hadoop-3.1.1.tar.gz
    mv hadoop-3.1.1/* /opt/hadoop/

    # Archivo de configuracion de las variables de entorno
    cat config-files/config-bash.txt >> ~/.bashrc
    source ~/.bashrc

    # Modificacion del archivo de configuracion hadoop-env.sh
    # Respaldo del archivo hadoop-env.sh
    cp /opt/hadoop/etc/hadoop/hadoop-env.sh /opt/hadoop/etc/hadoop/hadoop-env.backup
    # Limpieza del archivo
    # echo "" > /opt/hadoop/etc/hadoop/hadoop-env.sh
    # Sustitucion por el archivo con la variable de entorno JAVA_HOME con su valor correcto
    cp config-files/hadoop-env-config.sh /opt/hadoop/etc/hadoop/hadoop-env.sh

    # Modificacion del archivo de configuracion core-site.xml
    # Respaldo del archivo core-site.xml
    cp /opt/hadoop/etc/hadoop/core-site.xml /opt/hadoop/etc/hadoop/core-site.backup
    # Limpieza del archivo
    # echo "" > /opt/hadoop/etc/hadoop/core-site.xml
    # Sustitucion por el archivo con la configuracion correcta
```

```
cp config-files/core-site-config.xml /opt/hadoop/etc/hadoop/core-site.xml

# Modificacion del archivo de configuracion hdfs-site.xml
# Respaldo del archivo hdfs-site.xml
cp /opt/hadoop/etc/hadoop/hdfs-site.xml /opt/hadoop/etc/hadoop/hdfs-site.backup.xml
# Limpieza del archivo
# echo "" > /opt/hadoop/etc/hadoop/hdfs-site.xml

# Sustitucion por el archivo con la configuracion correspondiente
cp config-files/hdfs-site-config.xml /opt/hadoop/etc/hadoop/hdfs-site.xml

# Configuracion de los archivos mapred y yarn.
./parser.sh

# Modificacion del archivo de configuracion mapred-site.xml
# Respaldo del archivo mapred-site.xml
cp /opt/hadoop/etc/hadoop/mapred-site.xml /opt/hadoop/etc/hadoop/mapred-site.backup.xml
# Limpieza del archivo
# echo "" > /opt/hadoop/etc/hadoop/mapred-site.xml
# Sustitucion por el archivo con la configuracion correspondiente
cp config-files/mapred-site-config.xml /opt/hadoop/etc/hadoop/mapred-site.xml

# Modificacion del archivo de configuracion yarn-site.xml
# Respaldo del archivo yarn-site.xml
cp /opt/hadoop/etc/hadoop/yarn-site.xml /opt/hadoop/etc/hadoop/yarn-site.backup.xml
# Limpieza del archivo
# echo "" > /opt/hadoop/etc/hadoop/yarn-site.xml
# Sustitucion por el archivo con la configuracion correspondiente
cp config-files/yarn-site-config.xml /opt/hadoop/etc/hadoop/yarn-site.xml

# Configuracion del archivo workers
contador=1
while read line
do
    if [ $contador -ne 1 ]
    then
        echo $line >> /opt/hadoop/etc/hadoop/workers
    fi
    contador=$((contador + 1))
done < ips.txt

# Se escribe un 9 en el log de progreso de instalacion.
# El 9 corresponde a la configuracion de Hadoop de manera satisfactoria en el maestro
echo "9" > luminus-ins.log
instalar_hadoop_esclavos
}

function instalar_hadoop_esclavos() {
    # Copiado de los archivos de configuracion de hadoop del nodo maestro a los nodos
    contador=1
    while read line
    do
        if [ $contador -ne 1 ]
        then
```

```
echo "Instalando Hadoop en nodo $contador ($line)..."  
scp hadoop-3.1.1.tar.gz root@$line:/home/  
ssh -qq root@$line < hadoop-ins.sh  
scp config-files/core-site-config.xml root@$line:/opt/hadoop/etc/  
scp config-files/hadoop-env-config.sh root@$line:/opt/hadoop/etc/  
scp config-files/hdfs-site-config.xml root@$line:/opt/hadoop/etc/  
scp config-files/mapred-site-config.xml root@$line:/opt/hadoop/etc/  
scp config-files/yarn-site-config.xml root@$line:/opt/hadoop/etc/  
scp /opt/hadoop/etc/hadoop/workers root@$line:/opt/hadoop/etc/hadoop  
contador=$((contador + 1))  
fi  
contador=$((contador + 1))  
done < ips.txt  
  
# Se escribe un 10 en el log de progreso de instalacion.  
# El 10 corresponde a la configuracion e instalacion de Hadoop en los nodos esclavos  
# de manera satisfactoria en el maestro.  
echo "10" > luminus-ins.log  
}  
  
echo "***** Bienvenido a Luminus *****"  
  
pedir_ip_maestro  
instalar.hadoop_maestro  
  
## Ciclo para recolectar la informacion del archivo log  
# contador=1  
# while read line  
# do  
#     if [ $contador -eq 1 ]  
#     then  
#         line=$line  
#         break  
#     fi  
# done < luminus-ins.log  
  
# echo $line  
  
## Switch para saber en que paso iniciara el instalador  
# if [ $line -eq '1' ] || [ $line -eq '10' ]  
# then  
#     pedir_ip_maestro  
# fi  
# if [ $line -eq '2' ]  
# then  
#     echo "REANJUNDANDO INSTALACION..."  
#     pedir_ips_esclavos  
# fi  
# if [ $line -eq '3' ]  
# then  
#     echo "REANJUNDANDO INSTALACION..."  
#     instalar_java_maestro  
# fi  
# if [ $line -eq '4' ]
```

```
# then
#     echo "REANJUNDANDO INSTALACION..."
#     instalar_java_esclavos
# fi
# if [ $line -eq '5' ]
# then
#     echo "REANJUNDANDO INSTALACION..."
#     instalar_scala
# fi
# if [ $line -eq '6' ]
# then
#     echo "REANJUNDANDO INSTALACION..."
#     instalar_spark
# fi
# if [ $line -eq '7' ]
# then
#     echo "REANJUNDANDO INSTALACION..."
#     configurar_spark_maestro
# fi
# if [ $line -eq '8' ]
# then
#     echo "REANJUNDANDO INSTALACION..."
#     instalar_hadoop_maestro
# fi
# if [ $line -eq '9' ]
# then
#     echo "REANJUNDANDO INSTALACION..."
#     instalar_hadoop_esclavos
# fi
```

Código de scala-ins.sh

```
# Instalacion de Scala
wget https://downloads.lightbend.com/scala/2.13.0-M1/scala-2.13.0-M1.deb
dpkg -i scala-2.13.0-M1.deb
scala -version
```

Código de spark-ins.sh

```
# Instalacion de Spark
rm spark-2.1.0-bin-hadoop2.7.tgz
wget http://d3kbcqa49mib13.cloudfront.net/spark-2.1.0-bin-hadoop2.7.tgz
mkdir /opt/spark
tar -xzvf spark-2.1.0-bin-hadoop2.7.tgz
cp -a spark-2.1.0-bin-hadoop2.7/. /opt/spark/
rm -rf spark-2.1.0-bin-hadoop2.7
echo 'export SPARK_HOME=/opt/spark/' >> ~/.bashrc
echo 'export PATH="/opt/spark/bin:/opt/spark/sbin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

Código de hadoop-ins.sh

```
# Inicia instalacion de hadoop
# Descarga, descompresion y movimiento del
# contenido de hadoop a su carpeta

# wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-3.1.1/hadoop-3.1.1.tar.gz
rm -rf /opt/hadoop
```

```
mkdir /opt/hadoop  
tar -xzvf /home/hadoop-3.1.1.tar.gz  
mv hadoop-3.1.1/* /opt/hadoop/
```

Código de backup.sh

```
# Se ejecuta en el nodo maestro.  
cp /etc/hosts /etc/hosts.back  
cp ~/.bashrc ~/bashrc.back  
  
# Se ejecuta en cada uno de los nodos.  
contador=1  
while read line  
do  
    if [ contador -ne 1 ]  
        ssh root@$line "cp /etc/hosts /etc/hosts.back"  
        ssh root@$line "cp ~/bashrc ~/bashrc.back"  
    then  
        contador=$((contador + 1))  
    fi  
done
```

Código de parser.sh

```
#!/usr/bin/env bash  
function parsear_conjunto_valores_hdp() {  
rm -rf valores_hdp.txt  
while IFS='=' read -ra line; do  
    for i in "${line[@]}"; do  
        while IFS=' ' read -ra value; do  
            for j in "${value[1]}"; do  
                echo $value >> valores_hdp.txt  
            done  
        done <<< $i  
    done  
done <<< $(echo $(python hdp_conf_files/scripts/yarn-utils.py -c 1 -m 1 -d 1 false))  
}  
function generar_property() {  
    echo "<property>" >> $3  
    echo "<name>$1</name>" >> $3  
    echo "<value>">$(sed $2'q;d' valores_hdp.txt)</value>" >> $3  
    echo "</property>" >> $3  
}  
  
# Metodo que construye el archivo mapred-site-config.xml.  
function generar_mapred_site_xml() {  
    cp config-files/mapred-site-config.back.xml config-files/mapred-site-config.xml  
    # Property para el valor yarn.app.mapreduce.am.command-opts  
    # generar_property 'yarn.app.mapreduce.am.command-opts' '23' 'config-files/mapred-site-config.xml'  
    # Property para el valor mapred.map.memory.mb  
    generar_property 'mapreduce.map.memory.mb' '18' 'config-files/mapred-site-config.xml'  
    # Property para el valor mapreduce.reduce.memory.mb  
    generar_property 'mapreduce.reduce.memory.mb' '20' 'config-files/mapred-site-config.xml'  
    # Property para el valor yarn.app.mapreduce.am.resource.mb  
    generar_property 'yarn.app.mapreduce.am.resource.mb' '22' 'config-files/mapred-site-config.xml'  
    # Property para el valor mapreduce.map.java.opts  
    # generar_property 'mapreduce.map.java.opts' '19' 'config-files/mapred-site-config.xml'
```

```
# Property para el valor mapreduce.java.opts
# generar_property 'mapreduce.reduce.java.opts' '21' 'config-files/mapred-site-co
# Property para el valor mapreduce.task.io.sort.mb
# generar_property 'mapreduce.task.io.sort.mb' '24' 'config-files/mapred-site-co

        echo '</configuration >' >> config-files/mapred-site-config.xml
}

# Metodo que construye el archivo yarn-site-config.xml
function generar_yarn_site_xml() {
    cp config-files/yarn-site-config.back.xml config-files/yarn-site-config.xml
    # Property para el valor yarn.nodemanager.resource.memory-mb
    generar_property 'yarn.nodemanager.resource.memory-mb' '17' 'config-files/yarn-si
    # Property para el valor yarn.scheduler.maximum.allocation-mb
    generar_property 'yarn.scheduler.maximum.allocation-mb' '16' 'config-files/yarn-si
    # Property para el valor yarn.scheduler.minimum.allocation-mb
    generar_property 'yarn.scheduler.minimum.allocation-mb' '15' 'config-files/yarn-si
    # Property para el valor yarn.app.mapreduce.am.command-opts
    # generar_property 'yarn.app.mapreduce.am.command-opts' '23' 'config-files/yarn-si
    echo '</configuration >' >> config-files/yarn-site-config.xml
}

parsear_conjunto_valores_hdp
generar_yarn_site_xml
generar_mapred_site_xml
```