Luminus

Escuela Superior de Cómputo del IPN
Escuela Superior de Cómputo, IPN

20 de octubre de 2018

Índice general

CAPÍTULO 1

Capitulo 1

1.1. Introduccion

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. Como el propio término "Big Data"(Grandes Datos en español) lo hace entender, los la cantidad de datos que se maneja es demasiado grande para ser siquiera almacenada o procesada de manera tradicional. Big Data provee a quien lo usa conocimiento que de otra forma le habría sido imposible obtener, por lo que puede llegar a ser una herramienta de alto valor entre las empresas.

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. 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. Como son:

- Datos de gran volumen.
- Los datos cambian rápidamente por lo que su tiempo de validez es muy corto.
- En ocasiones es difícil encontrar patrones para agrupar datos provenientes de diversas fuentes.
- Es necesario discernir entre qué datos son importantes en el análisis y cuáles no lo son.
- Es necesario integrar diferentes tipos de datos (estructurados, no estructurados, semiestructurados).
- No existen estándares de calidad de datos unificados.

El problema con el uso de Big Data es que 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 algo pronunciadas, por lo que generalmente conlleva un tiempo algo 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, es bastante alto.

1.3. Propuesta de solución

En este trabajo terminal se propone desarrollar un ambiente de Big Data que incluya procesos de verificación de datos, selección, aplicación de algoritmos de minería de datos (clasificación y reglas de asociación) y machine learning necesarios, así como la generación de conocimiento y resultados a partir de la aplicación de esos algoritmos. Esta solución podrá implementarse en computadoras tradicionales, bajando así el costo económico de su implementación en gran medida, ya que se estaría eliminando la necesidad de comprar equipos especializados y ambientarlos.

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 algoritmos de minería de datos implementados.

SOFTWARE	CARACTERÍSTICAS	PRECIO EN EL MERCADO
Luminus	-Trabaja con datos que se encuentra en distintos equipos de cómputo (Red distribuida) -Se 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 -Ejecuta flujos de datos directamente hacia Spark	Próximamente en el mercado
Knime	-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. Tamaño de instalación: 1.98GB para la versión gratuita	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	-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 Tamaño de instalación: 1.4GB para la versión de prueba	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	-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	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.	-Basic - $0.07 USD por hora de uso de unidad de procesamiento. -Analytics - $0.2 USD por hora de uso de unidad de procesamiento.
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.	-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
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.	Gratuito

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 tiene la creencia de que es 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. 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.

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 (las cuales ya se han mencionado en los apartados previos) 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. 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. Además, el costo de las herramientas comerciales más robustas puede llegar a los cientos de miles de pesos.

Para la realización de esta propuesta se requerirá dominio en el manejo de bases de datos relacionales y no relacionales; conocimiento de técnicas de minería de datos y su adecuada aplicación; manejo de herramientas y Frameworks dedicados al desarrollo de aplicaciones que se ejecuten de manera distribuida; conocimiento en el ámbito de complejidades algorítmicas para así poder justificar la mejora en la eficiencia de esta herramienta con respecto a las otras ya existentes; manejo de distintos lenguajes de programación, debido a que el lenguaje que se utilizara para el desarrollo de la aplicación principal será distinto al lenguaje que se utilizara para la implementación de los algoritmos de minería de datos.

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 (Framework) [9] 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. Especificos

- 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

Desarrollo de un software que permita la implementación de un ambiente de Big Data de manera distribuida para realizar análisis de datos aplicando diversas técnicas de minería de datos como son: aprendizaje de máquina (machine learning), como reglas de asociación y árboles de decisión mismos que favorecen de una u otra manera a la toma de decisiones de forma práctica y sin demasiadas complicaciones, a las empresas. El siguiente diagrama muestra la arquitectura prevista del sistema:

Con esta arquitectura será posible adaptar el Framework a diversos casos de estudio relacionados a datos de in-

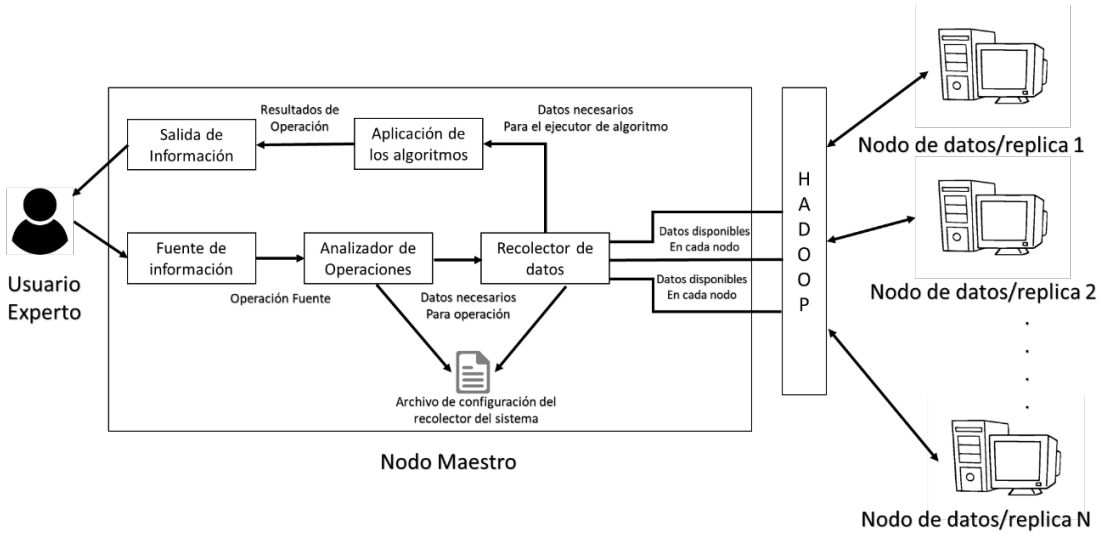


Figura 1.1: Arquitectura de la red distribuida

gresos y egresos económicos para diferentes 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.

Productos esperados:

1. Framework funcional para el manejo de Big Data en grandes cantidades de datos. (Implementación, Código Fuente)
2. Caso de prueba funcional implementado en el Framework
3. Manual Técnico
4. Manual de Usuario.

1.8. Alcances y 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:

- 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 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 los alcances del proyecto son limitados por lo que se describirá un poco acerca de ello

- Solo se tendrán 3 algoritmos de minería de datos para ejecutar dentro del Framework, 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 unicamente 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 se suban a el repositorio , tendrán que subirse por separado y el manejo de los datos que se podrá ejecutar descargara 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 otro tipo de equipos de computo 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.

1.9. Trabajo a futuro

1.9.1. Prototipo 3: Página web y algoritmos de minería de datos

Se pretende desarrollar este prototipo en Trabajo Terminal 2 el cual incluirá las siguientes tareas:

- Hacer uso de un algoritmo de cada una de los tipos de algoritmos que se enlistan a continuación:
  - Reglas de asociación
  - Arboles de decisión
  - Aprendizaje maquina

escrito en java que se adapte al caso de estudio así como su personalización para que pueda ser adaptado a otros casos de estudio de manera sencilla.

- Elaboración de las pantallas que conformarán el sitio web, diseño, estructura y contenido que se puede ver como un mapa de navegación que aun no esta integrado a Luminus por lo tanto, no es funcional.

1.9.2. Prototipo 4: Algoritmos de minería de datos desde una pagina web

Se pretende desarrollar este prototipo en Trabajo Terminal 2 el cual comprende el desarrollo de la siguiente tarea:

Integrar los 2 puntos comprendidos en el prototipo 3 los cuales pueden ser consultados en la sección anterior 1.9.1 de tal forma que los algoritmos de minera de datos puedan ser accedidos y utilizados desde la pagina web elaborada.
Además de que estos algoritmos se ejecutarán en el cluster generado en los prototipos Prototipo 1 Evaluación y definición de requerimientos de la red distribuida y Prototipo 2 Integración de requerimientos de la red distribuida desarrollados en trabajo terminal 1.

1.9.3. Prototipo 5: Luminus como framework

Este prototipo se encontraba diseñado unicamente para convertir este proyecto a un framework que pueda ser adaptado a otros casos de estudio, sin embargo, se vio la necesidad de crear un instalador que permita cargar todo el contenido del Trabajo Terminal al equipo del cliente para que este pueda utilizarlo.
Esto con el objetivo de que, el instalador cargue a su equipo todo lo que Luminus necesita para funcionar. y unicamente trabaje sobre el Framework.
por tal motivo se explicarán los alcances del Framework y del instalador por separado.

Framework

Para la parte del Framework se busca que se diseñe una plantilla de ajuste la cual permita la adaptación para otros casos de estudio debido a que los algoritmos de Big Data suelen tener diferentes entradas, las cuales cambian de acuerdo para cada caso de estudio.

Instalador

Un instalador suele ser un sistema bastante complejo, ya que realiza las siguientes tareas:

- Instalación de software
- Ejecuta comandos en el shell de la computadora
- Verifica software que ya está instalado en la computadora y generalmente es capaz de cambiar la versión

Se tiene una parte del instalador desarrollada en Trabajo Terminal 1 la cual carece de robustez, por lo que se pretende que durante el desarrollo del Trabajo Terminal 2, este se vuelva más robusto de lo que es ahora, es decir:

- Que sea capaz de reanudar el proceso de instalación cuando éste haya sido interrumpido. Ya sea por decisión del usuario o por un error en el proceso de instalación. El instalador, debe retomar el proceso a partir del punto en el que fue interrumpido.
- Que pueda agregar nodos a la red y remover nodos de la red.
- Que sea capaz actualizar las IPs de los nodos cuando exista algún cambio en ellas.
- Que pueda identificar si alguna de las tecnologías requeridas para la instalación de Luminus, ya existe en alguno de los nodos. De ser así, se validará la versión de la tecnología en cuestión. Si la versión es diferente a la que necesita Luminus, se pregunta al usuario si desea conservar su versión del software, o la versión necesaria para Luminus. Si la versión del software previamente instalada es la correcta, se notifica que Luminus no instaló esa tecnología.

Por otro lado, se agregarán al instalador todos los prototipos contenidos en el trabajo terminal.

CAPÍTULO 2

Marco teórico

2.1. Marco Teórico

2.1.1. Big Data

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.

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 el crecimiento de la nececidad de almacenamiento de datos en el mundo moderno, crece de forma exponencial.

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.

3. **Variedad:** Procesamiento de datos de naturaleza heterogénea, es decir, múltiples tipos de datos.

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.

• **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.

• **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.

• **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.

2.1.2. 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.

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.

• **Riesgo y probabilidad:** Es un escenario muy común dentro de los negocios de Bussiness 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.

• **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.

• **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.

• **Agrupación:** Clasificación de los elementos de un conjunto de información para el posterior análisis de sus afinidades.

2.1.3. 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.

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:
   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.

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.

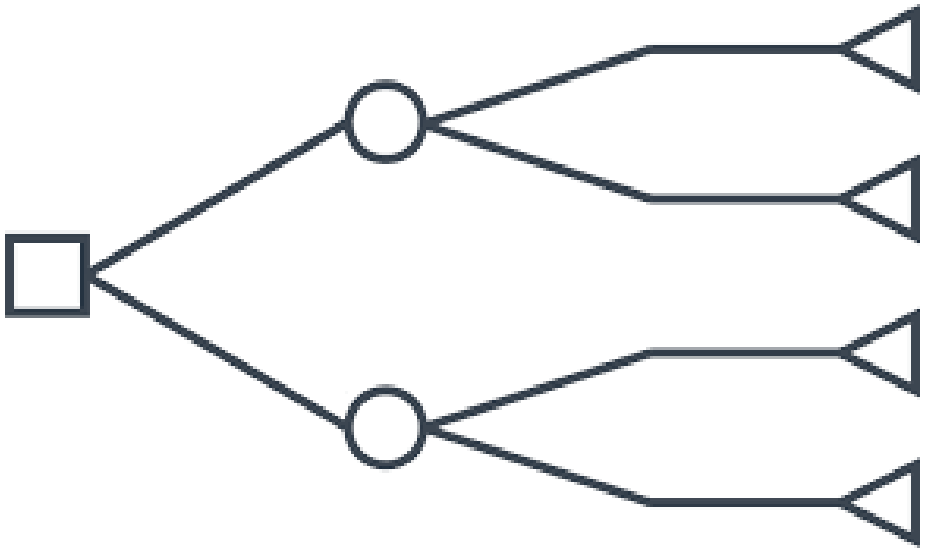


Figura 2.1: Representación general de un árbol de decisión

2.1.4. 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.

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

Entropía

La entropía es la medida de la aleatoriedad. En otras palabras, la medidad 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 comienca 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 siguiente:

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

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.

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 debería jugarse:

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*.

- Después 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.

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
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

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

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

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

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

Cuadro 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 demas colúmnas/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

Cuadro 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

Cuadro 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

Cuadro 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

Cuadro 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

Cuadro 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

Cuadro 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

Cuadro 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:

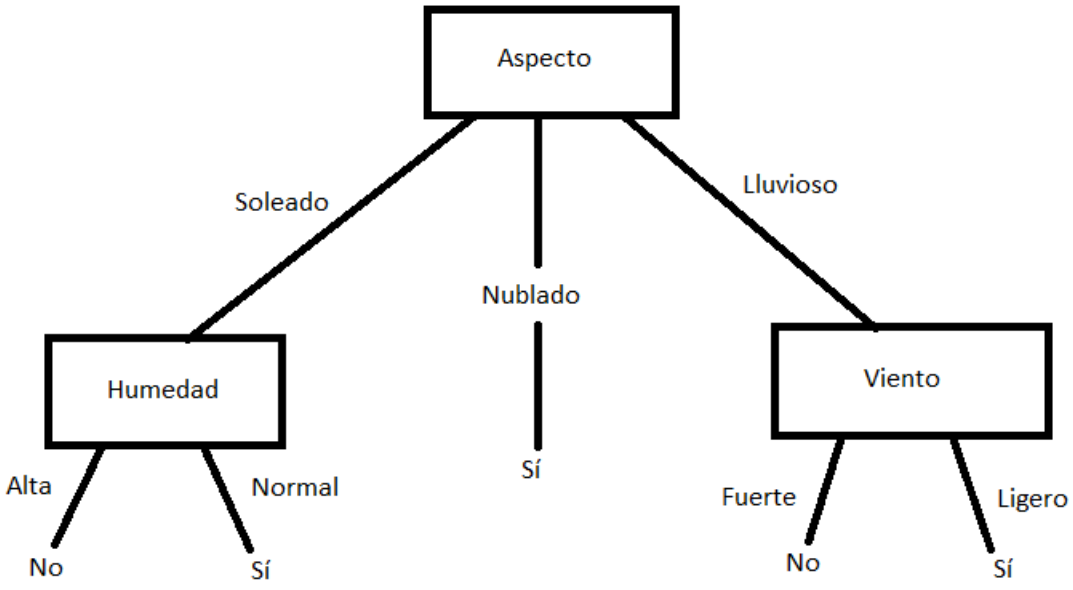


Figura 2.4: Árbol de decisión resultante.

2.1.5. C4.5

El algoritmo C4.5 es la evolución del algoritmo ID3. Éste genera un árbol de decisión a partir de un conjunto de dstos de entrada de manera recursiva, al igual que su precursor. 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:

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

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

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

Cuadro 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:

1. 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

Cuadro 2.12: Tabla de l 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 <>65) = 0.048
   1.4. SplitInfo(Decisión, Humedad <>65) = 0.371
   1.5. GainRatio(Decisión, Humedad <>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 <>70) = 0.014

2.4. SplitInfo(Decisión, Humedad <>70) = 0.863

2.5. GainRatio(Decisión, Humedad <>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 <>75) = 0.045

3.4. SplitInfo(Decisión, Humedad <>75) = 0.940

3.5. GainRatio(Decisión, Humedad <>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 <>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

Cuadro 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

Cuadro 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

Cuadro 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

Cuadro 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:

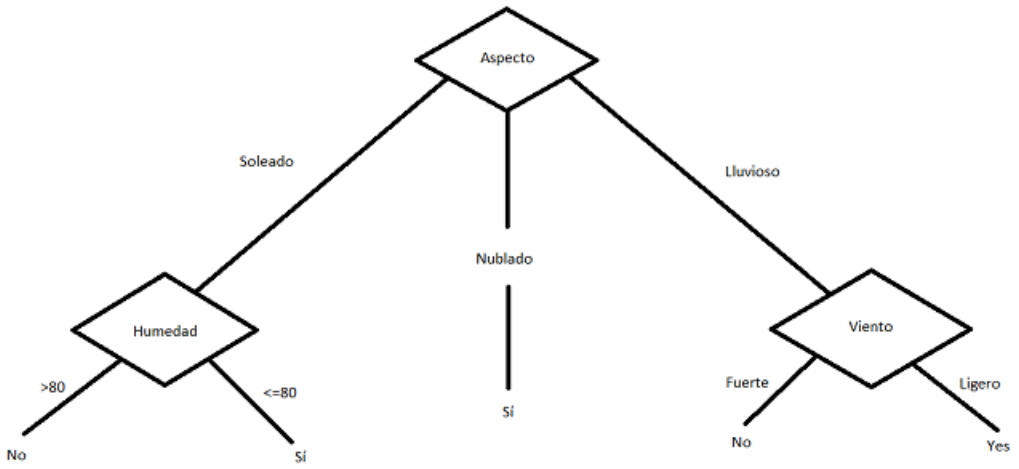


Figura 2.7: Arbol de decisión construido con C4.5.

2.1.6. Algoritmo KNN (K-Nearest Neighbor)

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.

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

1. Se calcula la similitud entre puntos basaándose en una función de distancia.

2. Se encuentran los K vecinos más cercanos.

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

Cuadro 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 Euclidiana.

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

Cuadro 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

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

2.1.7. 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. 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.

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

Cuadro 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)

Cuadro 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)

Cuadro 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)

Cuadro 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)

Cuadro 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

Cuadro 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)

Cuadro 2.26: Partición final.

- Las iteraciones continuarían hasta que ya no haya más reubicaciones.

- Es probable que el algoritmo no encuentre una solución final.

2.1.8. 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.

2.1.9. Hadoop

Es 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.

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 paalelizar tareas de procesamiento de datos.

- **Eficiencia:** Mediante la paralelización, se consigue una ganancia considerable en el tiempo de procesamiento de información.

- **Económico:** Propicia un ambiente fácilmente escalable en el que resulta sencillo añadir nodos de manera horizontal conforme se vaya requiriendo.

- **Código abierto:** Es un proyecto de los llamados **open source**.

- **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.

Arquitectura de Hadoop

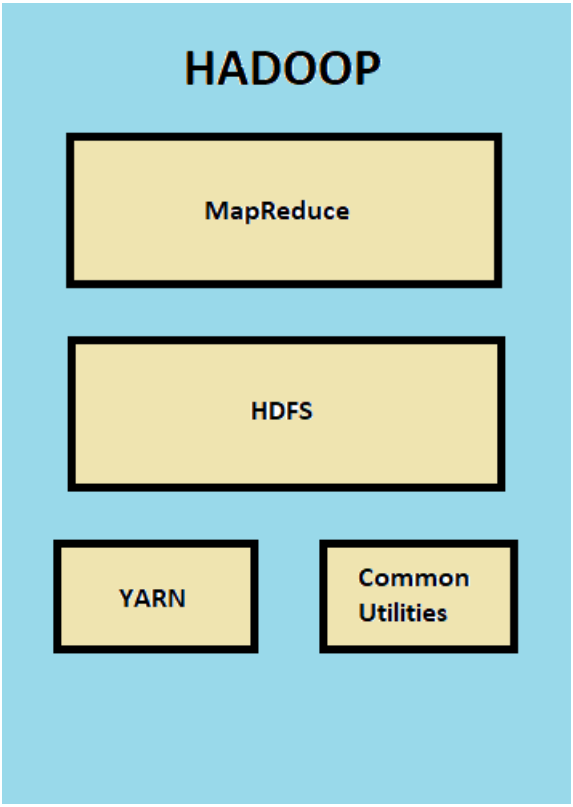


Figura 2.9: Arquitectura básica de Hadoop.

- **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.

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**.

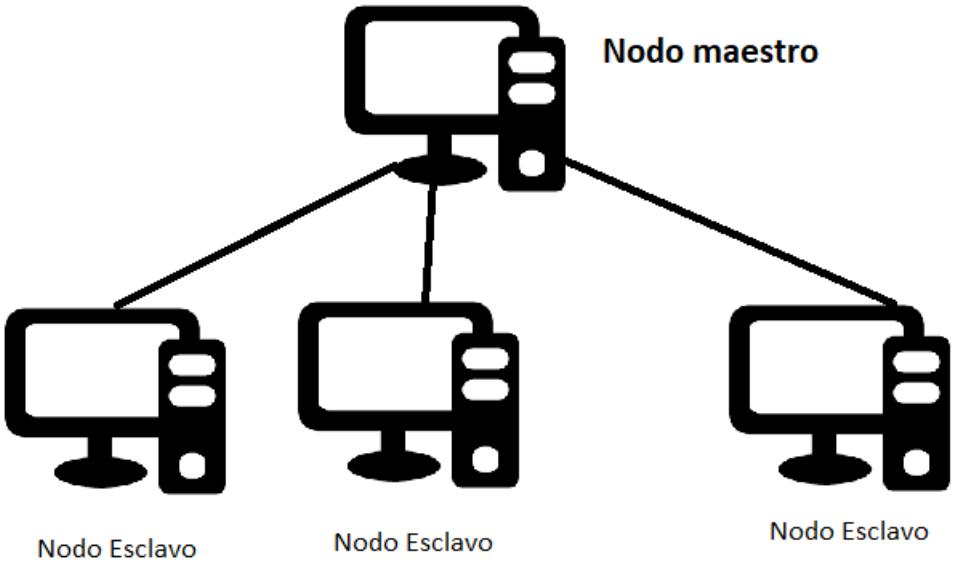


Figura 2.10: Arquitectura de maestros y esclavos.

Distribuciones Comerciales de Hadoop

Existen varias opciones actualmente en el mercado para hacer uso de las capacidades de Hadoop, entre las cuales destacan 3 principalmente:

- **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.

2.1.10. 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.

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.

Ventajas de usar Apache Spark

Utilizar Spark para el procesamiento de volúmenes de información de gran tamaño ofrece varios beneficios. Los principales beneficios son los siguientes:

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

Componentes de Apache Spark

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

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

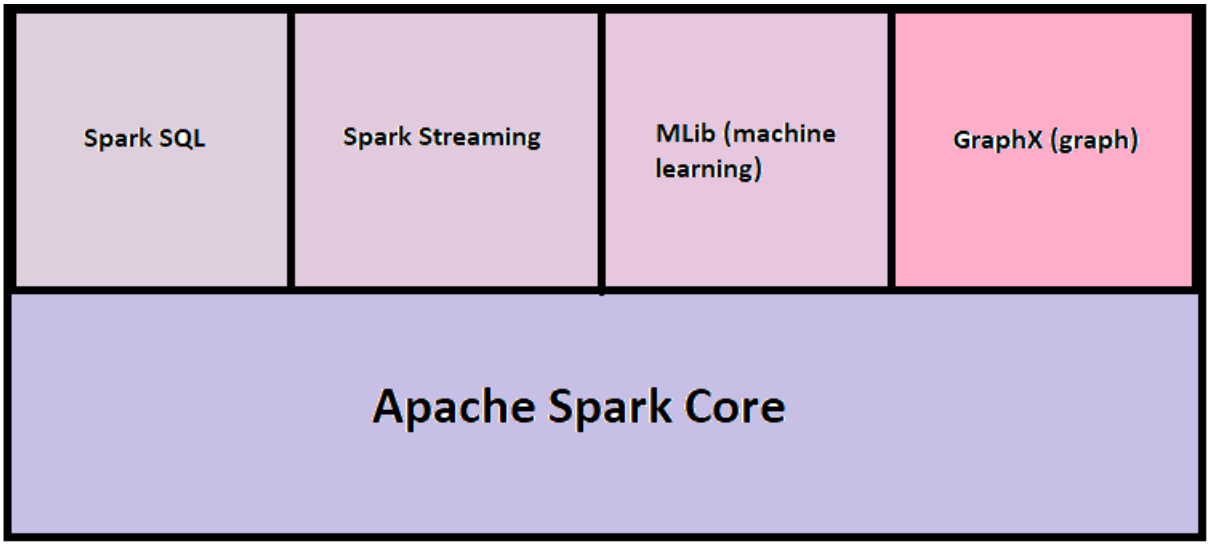


Figura 2.11: Componentes que conforman Apache Spark.

- **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.

- **MLib:** 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.

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 que son necesarios 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 esta 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 determino 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 mas 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 mas 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 sera 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 diseño 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 asigno 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 framework 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.

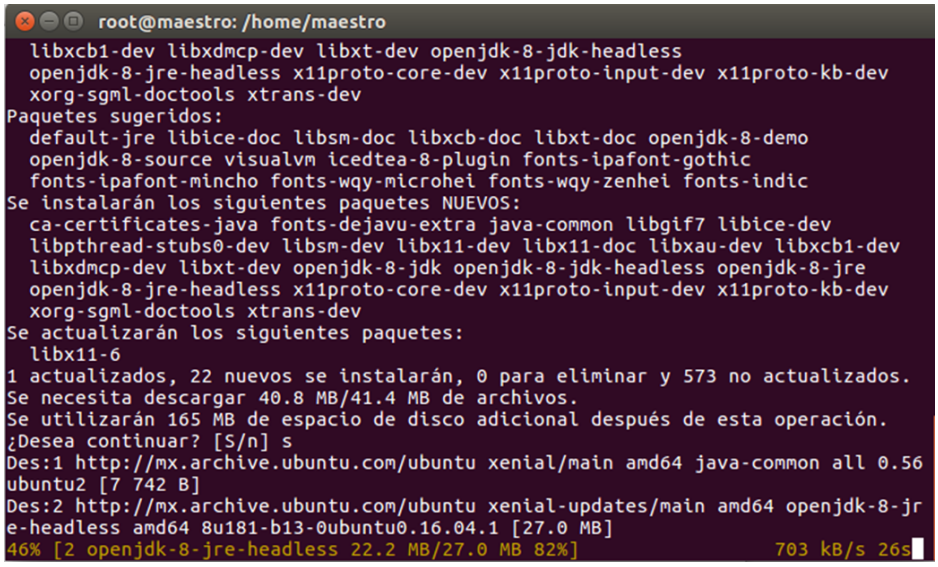


Figura 3.1: Arquitectura del sistema

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 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.

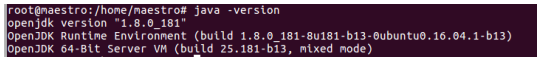


Figura 3.2: Generación de bloques de un archivo

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.

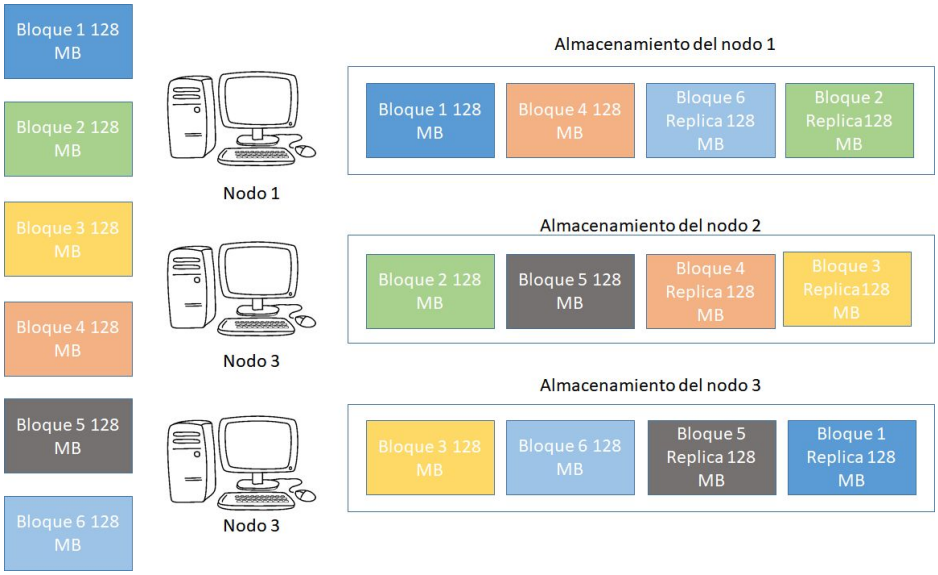


Figura 3.3: Generación de bloques de un archivo

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 sera 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

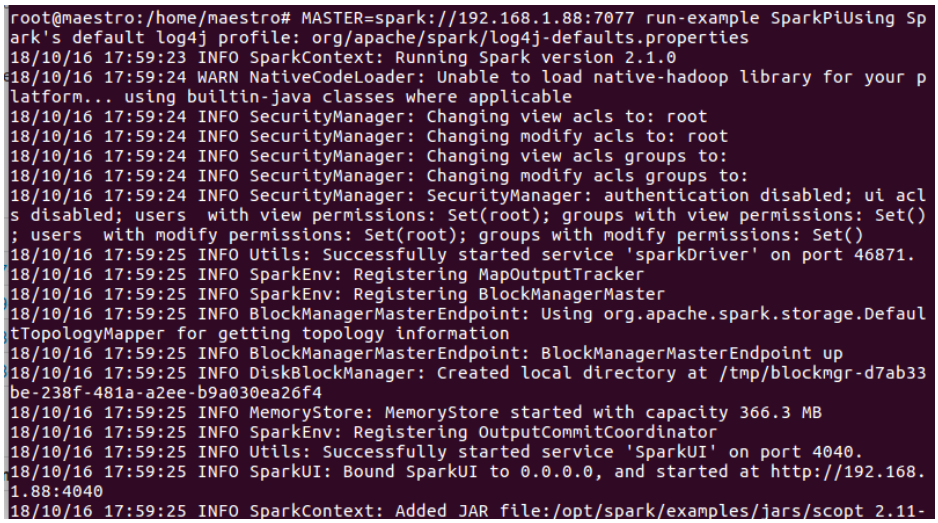


Figura 3.4: Ejecución de algoritmo SparkPi

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

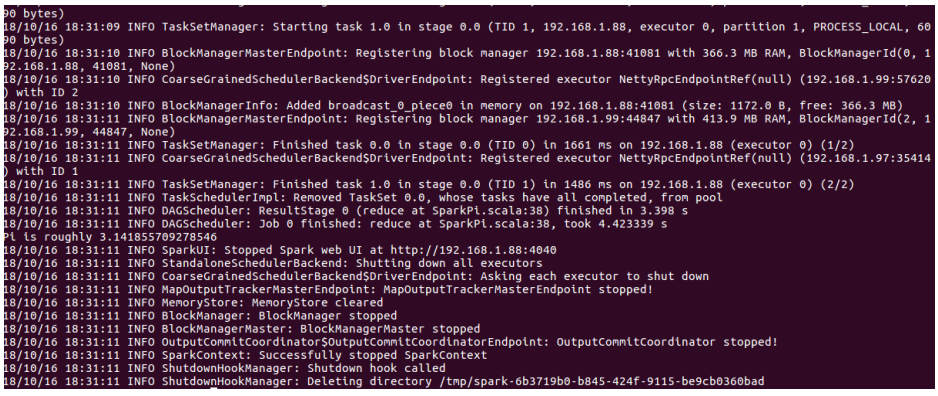
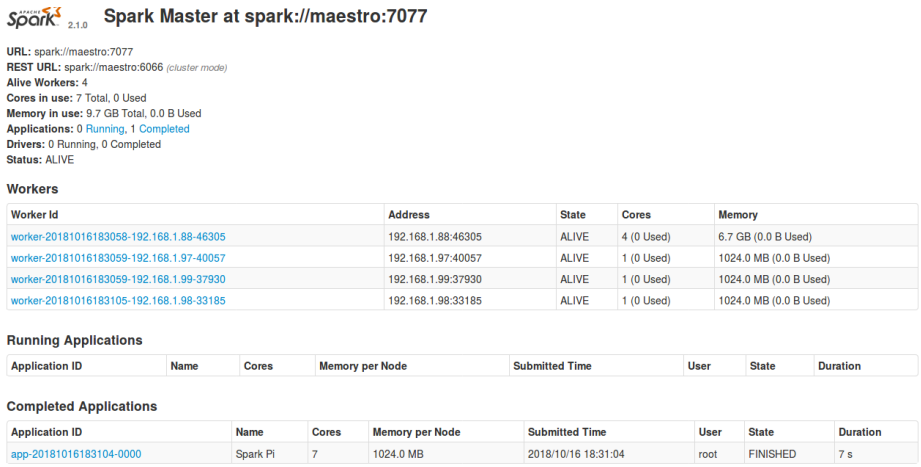


Figura 3.5: Valor de PI

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 sera 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 2.1.0 **Spark Master at spark://maestro:7077**

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

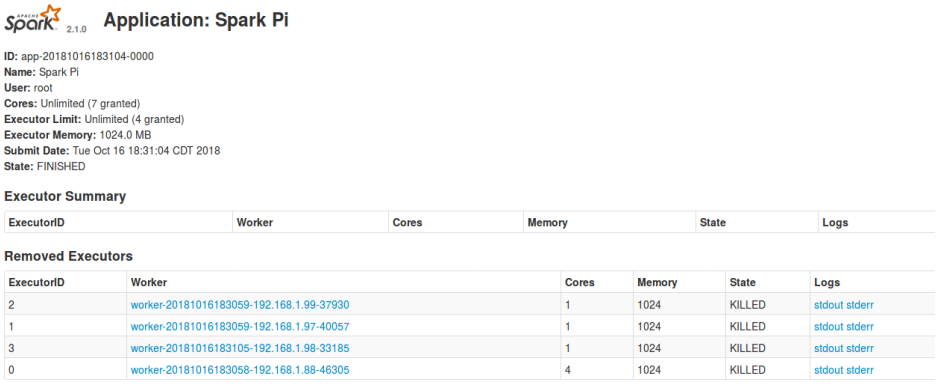
Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

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



Spark 2.1.0 **Application: Spark Pi**

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

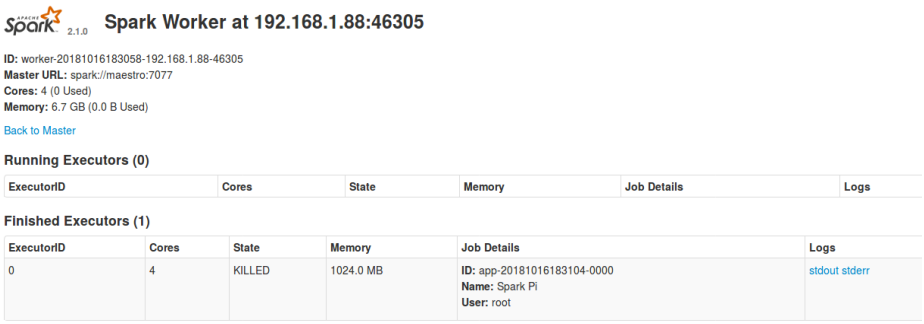
ExecutorID	Worker	Cores	Memory	State	Logs

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 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 capitulo 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

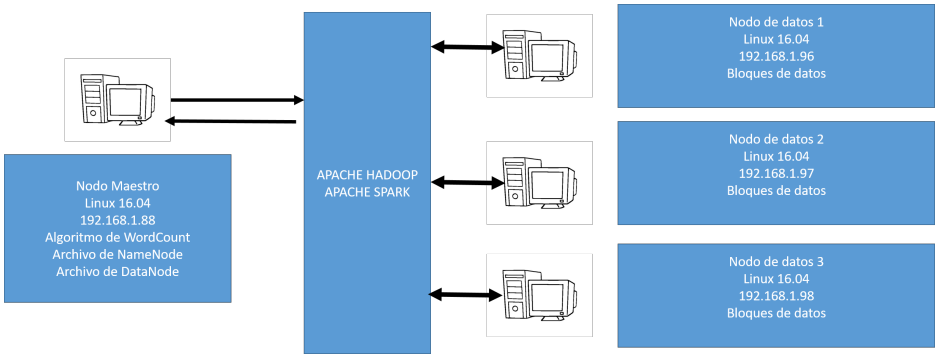


Figura 4.1: Diseño de la red distribuida con nodos de datos

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 utilizara 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 busco 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 Çontador de palabras.º "WordCount.ésteste 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 serian utilizados si se tratara de un algoritmo que se ejecuta sin el uso de Map Reduce.

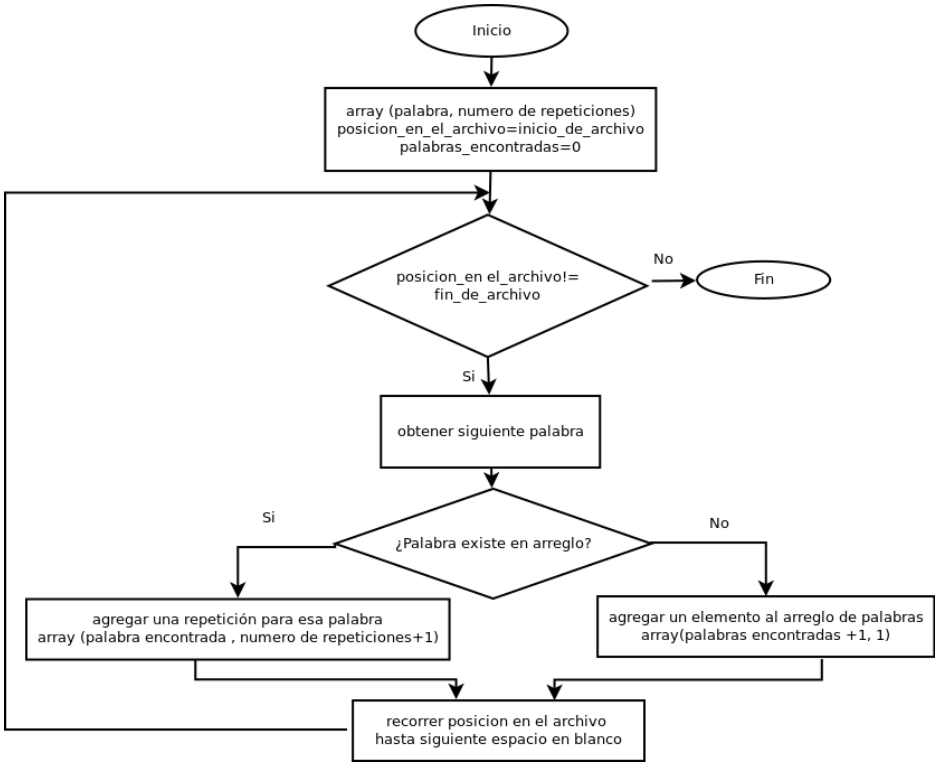


Figura 4.2: Diagrama de flujo del algoritmo contador de palabras

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 funcion Map en este ejemplo haría seria 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, Refresco ,RefrescO , botellarefresco .

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

(botella ,1), (refresco ,1), (Botella ,1), (BOTELLA,1),
(Refresco ,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 seria 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
(Refresco ,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

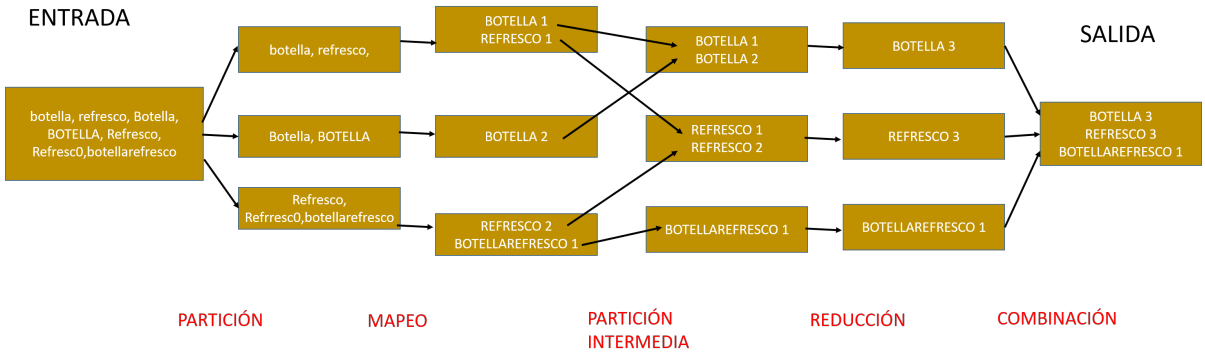


Figura 4.3: Funcionamiento completo de Hadoop, con todas sus operaciones

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.

1. **Partición**: Se requiere definir un parametro 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').

2. **Mapeo**: Se explico anteriormente en WordCount con el uso de Map Reduce `Función Map`.

3. **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.

4. **Reducción**: Se explico anteriormente en WordCount con el uso de Map Reduce `Función Reduce`.

5. **Combinación**: Todos los datos de salida que arrojo 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

```
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
6. Puesta en funcionamiento
6.1. 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 capitulo, 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 codigo JAVA para tal objetivo se puede visualizar en el **??** 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 pasarla 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 codigo 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 especifico 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

```
     nodo3:38667                RUNNING         nodo3:8042
               0
     nodo2:44891                RUNNING         nodo2:8042
               0
root@maestro:/opt/hadoop/etc/hadoop# yarn jar /home/maestro/Escritorio/MRProgr
amsDemo.jar PackageDemo.WordCount /user/luminus/datosproductos/all_data.csv /u
ser/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 proce
ss : 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.s
ystem-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metric
s-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 applic
ation_1539850756835_0001
2018-10-18 03:20:35,328 INFO mapreduce.Job: The url to track the job: http://m
aestro:8088/proxy/application_1539850756835_0001/
2018-10-18 03:20:35,329 INFO mapreduce.Job: Running job: job_1539850756835_000
1
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

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.

```
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 funcion 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 estadisticas 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 complet
ed 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

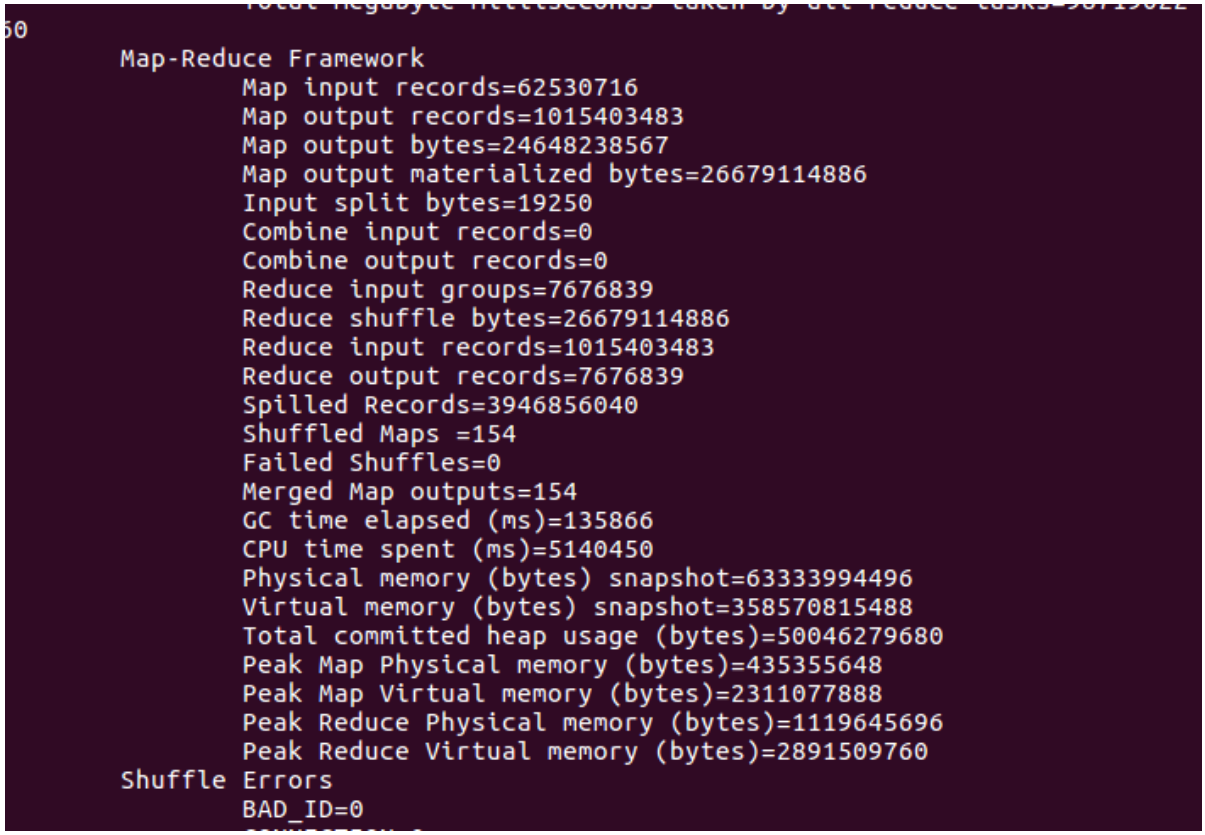


Figura 4.7: Ejecución de algoritmo map reduce para contar palabras: Estadísticas

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.

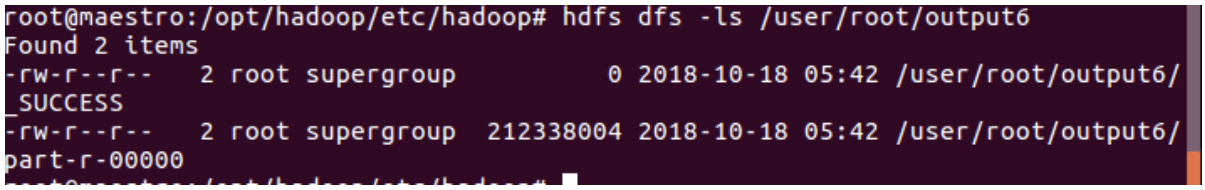


Figura 4.8: Contenido del directorio de salida

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

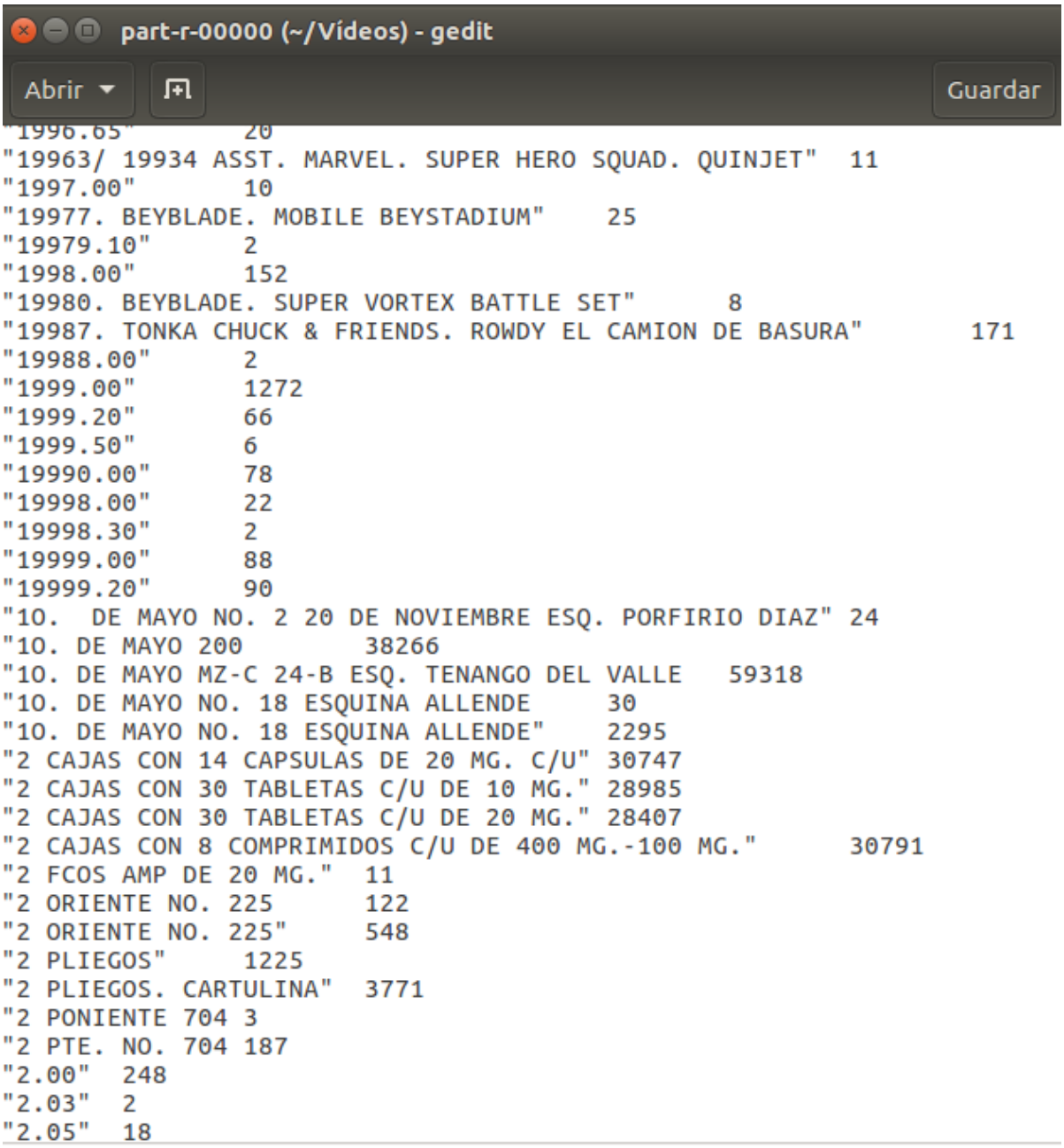


Figura 4.9: Contenido del archivo de resultados

no proporciona información real de los articulos y genera mucho ruido en el archivo.
Existen mas entradas de fechas que información de los productos, comportamiento que dificultaría el análisis de los datos en un futuro por lo cual este atributo sera 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 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
- 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 de el

Por lo tanto, el prototipo dos concluye de manera exitosa

CAPÍTULO 5

Instalador de Luminus

5.1. Descripción del prototipo

En este punto del trabajo terminal, este prototipo se encuentra en una fase aún experimental, sin embargo logra el cometido de simplificar el proceso de puesta en marcha.

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 comenzó el desarrollo de un instalador que simplifique el proceso de puesta en marcha del ambiente de análisis de datos.

La manera en que el instalador simplifica 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.

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.

5.2. Construcción y funcionamiento del prototipo

Este prorotipo será ejecutado solamente desde la máquina queque sea el nodo maestro de la red distribuida. Deberá ser ejecutado. Al iniciar la ejecución del instalador, 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.

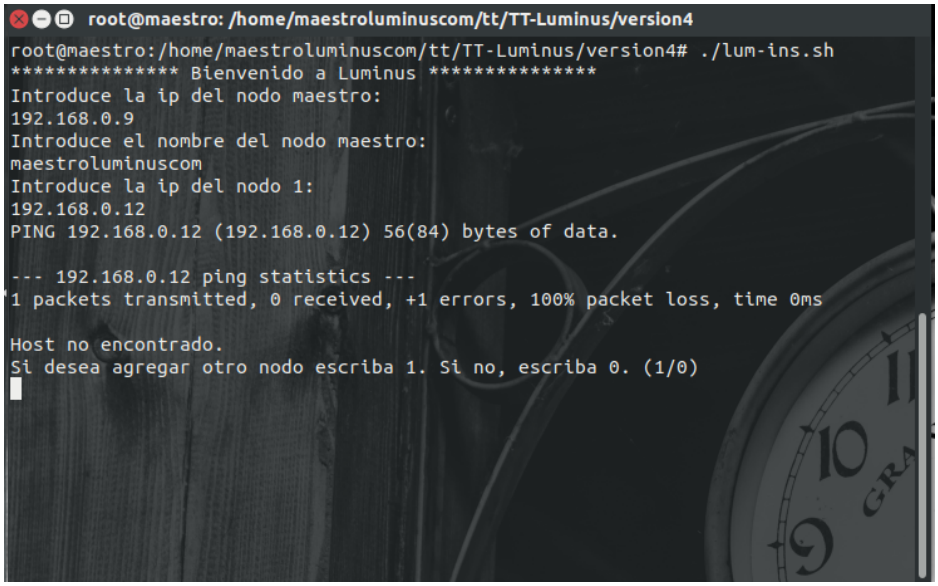


Figura 5.1: El usuario introdujo la IP de un host no encontrado.

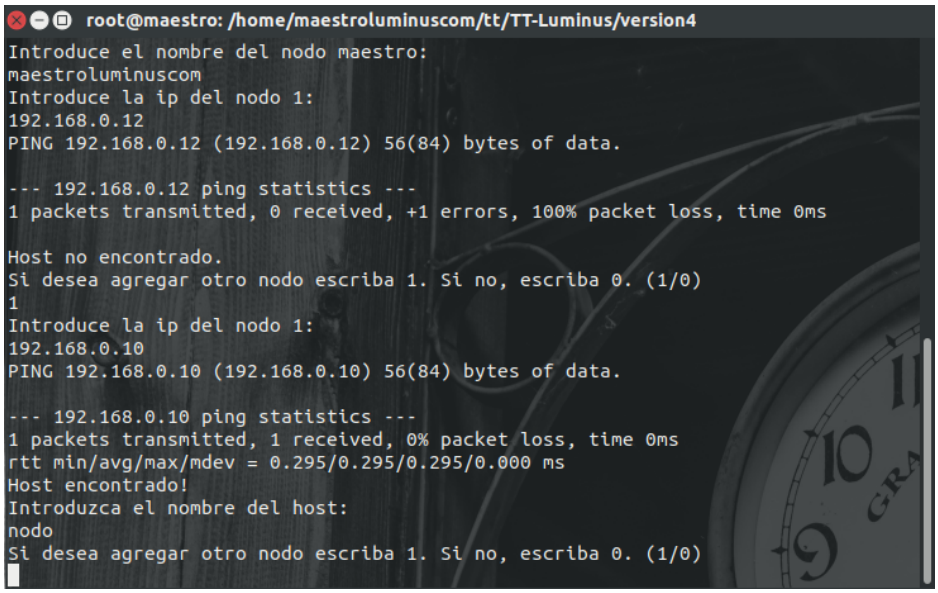


Figura 5.2: El usuario introdujo la IP de un host encontrado.

Posteriormente, cuando ya no se deseen agregar más nodos, el instalador ejecutará varios scripts de instalación en el nodo maestro, desde donde fue ejecutado.

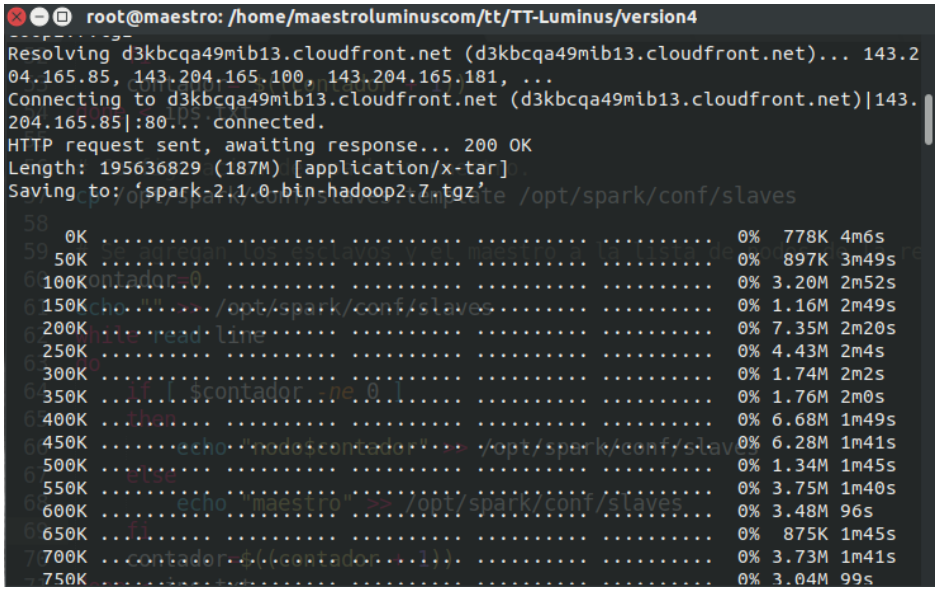


Figura 5.3: Comienza el proceso de instalación.

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

1. Java.

2. Scala.

3. 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 introdujo 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 demás nodos.

5.2.1. Diagrama de flujo

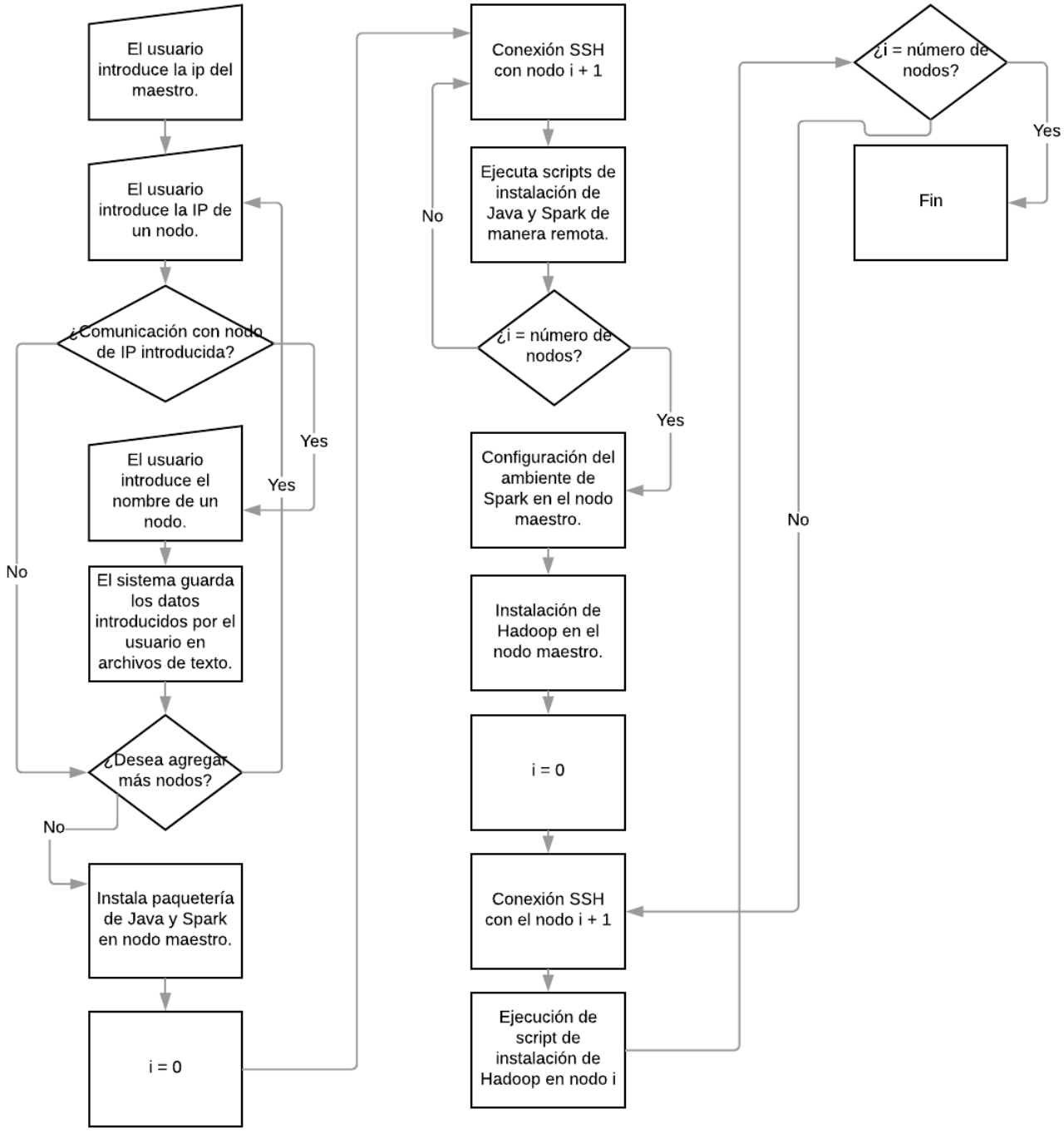


Figura 5.4: Diagrama de flujo de los pasos básicos que sigue el instalador.

Este prorotipo se realizó por completo utilizando la tecnología Shell Script en Ubuntu 18.04.

5.3. Alcances y trabajo a futuro

Un instalador suele ser un sistema bastante complejo, ya que realiza las siguientes tareas:

- Instalación de software
- Ejecuta comandos en el shell de la computadora
- Verifica software que ya está instalado en la computadora y generalmente es capaz de cambiar la versión

Para la realización de este trabajo terminal, se pretende que durante el desarrollo del Trabajo Terminal 2, este prorotipo se vuelva más robusto de lo que es ahora, es decir:

- Que sea capaz de reanudar el proceso de instalación cuando éste haya sido interrumpido. Ya sea por decisión del usuario o por un error en el proceso de instalación. El instalador, debe retomar el proceso a partir del punto en el que fue interrumpido.
- Que pueda agregar nodos a la red y remover nodos de la red.
- Que sea capaz actualizar las IPs de los nodos cuando exista algún cambio en ellas.
- Que pueda identificar si alguna de las tecnologías requeridas para la instalación de Luminus, ya existe en alguno de los nodos. De ser así, se validará la versión de la tecnología en cuestión. Si la versión es diferente a la que necesita Luminus, se pregunta al usuario si desea conservar su versión del software, o la versión necesaria para Luminus. Si la versión del software previamente instalada es la correcta, se notifica que Luminus no instaló esa tecnología.

CAPÍTULO 6

Anexos

6.1. Anexo A: Codigo 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));

                }

        }

}
```