Universidad de las Ciencias Informáticas

Facultad 4, GITAE

Entorno distribuido para el minado de datos en ambientes educacionales masivos.

Trabajo final presentado en opción al título de Máster en Informática Avanzada.

Autor: Ing. Neysa Baldoquin Alonso.

Tutores: Dr. C. Juan Pedro Febles Rodríguez

MSc. Angel Alberto Vazquez Sánchez.

Co-Tutores: MSc. Yuiesky Coca Bergolla.

.

La Habana, septiembre de 2018

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Neysa Baldoquin Alonso, con carné de identidad 92093022499, soy la autora principal del trabajo final de maestría Entorno distribuido para el minado de datos en ambientes educacionales masivos, desarrollada como parte de la Maestría en Informática Avanzada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año _____.

_____

Firma del autor

Ing. Neysa Baldoquin Alonso

RESUMEN

El desarrollo de las tecnologías de la información y las comunicaciones ha abierto una gama de posibilidades en el ámbito educativo. Uno de los mayores desafíos que afrontan las instituciones educativas es el crecimiento exponencial de los datos educativos y la utilización de los mismos para mejorar las decisiones administrativas. La aplicación de la minería de datos en la educación es concebida con el desarrollo de métodos para explorar los datos que provienen de entornos educativos. Para la exploración de dichos datos, existen técnicas novedosas que permiten realizar grandes procesamientos de cómputo utilizando hardware de pocas prestaciones y técnicas de minería de datos distribuida. Sobre esta base, existen modelos desarrollados para estimar y predecir cómo y cuánto aprenden los estudiantes, los cuales necesitan un entorno que brinde soporte a los procesos que se ejecutan dentro de estos modelos para garantizar el procesamiento de los datos de forma eficiente.

El objetivo de la presente investigación es desarrollar un entorno distribuido para contribuir al uso de técnicas de minería de datos distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos. Como resultado de la investigación que se llevó a cabo, se obtuvo un entorno distribuido capaz de aglutinar bases de datos heterogéneas que son utilizadas para el procesamiento eficiente de datos a través de técnicas de minería de datos educacionales. A través de las diversas pruebas realizadas, se validó la eficiencia con que cuenta dicho entorno para su explotación.

Palabras claves: Datos educativos, minería de datos, minería de datos educativos, entorno distribuido, eficiencia.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Uno de los mayores desafíos que afrontan las instituciones educativas es el crecimiento exponencial de los datos educativos y la utilización de los mismos para mejorar las decisiones administrativas (Cristobal Romero, 2013).

La Minería de Datos (MD) o *Data Minig* se plantea como un área del conocimiento que tiene como objetivo la aplicación de técnicas para procesar y analizar grandes volúmenes de información con los que se puede descubrir patrones repetitivos, tendencias, o reglas que expliquen el comportamiento de los datos en un determinado contexto (S.L. González-Ruiz, 2015).

La aplicación de minería de datos en la educación es un campo emergente de investigación interdisciplinaria también conocido como Minería de Datos Educativos (MDE) o *Educational Data Mining*. Es concebido con el desarrollo de métodos para explorar los tipos únicos de datos que provienen de entornos educativos. Es utilizada para entender mejor cómo los estudiantes identifican los entornos en los que aprenden, mejorar los resultados educativos y a comprender y explicar los fenómenos educativos. La información educacional generada puede almacenar una gran cantidad de datos potenciales de múltiples fuentes en diferentes formatos y con diferentes niveles de granularidad. Cada problema educativo particular tiene un objetivo específico con características especiales que requieren un tratamiento diferente del problema de la minería. Los problemas significan que las técnicas tradicionales de la MD no pueden aplicarse directamente a estos tipos de datos y problemas. Como consecuencia, el proceso de descubrir conocimiento tiene que ser adaptado y para ello se necesitan técnicas específicas de la MD (Cristobal Romero, 2013).

Los Ambientes Virtuales de Aprendizaje (AVAs) generan una gran cantidad de datos relacionados con actividades de los estudiantes. Esta información es generada cuando un estudiante se inscribe en un curso, el cual produce una gran cantidad de información referente al mismo, la cual es utilizada con el fin de monitorear características del curso. Comúnmente esta información generada es presentada en formato tabular, dependiendo de la cantidad o del tipo de datos, resulta difícil de interpretar. Este gran volumen de información no puede ser procesado e inspeccionado manualmente. La Minería de Datos Educacional (MDE) se basa en este precepto, ya que es muy apropiada para descubrir información en una base de datos de un AVA. Los datos recopilados son procesados en entornos *e-learning* teniendo como finalidad la identificación de patrones útiles aplicables a la evaluación de la actividad del usuario en la web y descubrir más profundamente cómo aprenden los estudiantes (Constanza R. Huapaya, 2012).

La MDE está basada en el desarrollo, investigación y aplicación de métodos para detectar patrones en colecciones grandes de datos educativos que de otro modo serían difíciles o imposibles de analizar (Cristobal Romero, 2013). La MDE es un área multidisciplinaria en la cual pueden incidir diversos métodos y técnicas de inteligencia artificial, como es el caso de la programación lógica, las redes neuronales artificiales, la lógica difusa, agrupamiento, probabilidades, clasificación, minería de textos, entre otros muchos (Constanza R. Huapaya, 2012). Esta nueva técnica emerge como un paradigma orientado a la generalización de modelos, tareas, métodos y algoritmos utilizados para la explotación de datos que provienen del contexto educativo, y tiene como función encontrar y

analizar patrones que sean capaces de caracterizar el comportamiento en base a los logros, evaluaciones y el dominio del contenido de conocimiento que demuestren los estudiantes a través de los diversos mecanismos de enseñanza-aprendizaje. Su objetivo es generar modelos educativos en los cuales se fomenten nuevas técnicas o herramientas que puedan analizar e incrementar el nivel participativo de los estudiantes. La recomendación de actividades para ofrecer nuevas experiencias de aprendizaje, avisos o predicciones del rendimiento de los alumnos para mejorar la efectividad del curso o promover el trabajo en grupo, es un ejemplo clave de utilización de estos mecanismos. Desde un punto de vista general se puede afirmar que MDE tiene como propósito incidir en el estudiante, donde el instructor/investigador adquiera un conocimiento y lo convierta en aprendizaje, mientras que el alumno, como el usuario final, se apropie del mismo llevándolo a un contexto de su vida cotidiana (Ballesteros Román & Daniel Sánchez-Guzmán, 2014).

Las técnicas de minería de datos utilizadas en el ámbito educativo, se han empleado con éxito para crear modelos predictivos en cuanto al rendimiento de los estudiantes, obteniendo a partir de los mismos resultados prometedores, que demuestran cómo determinadas características del entorno del alumno, pueden influir en sus resultados docentes (Carlos Márquez Vera, 2012).

Para la extracción de datos y procesamiento de los mismos son empleados diferentes tipos de algoritmos para la extracción del conocimiento tanto descriptivas (agrupamiento, correlaciones y reglas de asociación) como predictivas (la clasificación), además de algoritmos de selección y búsqueda de mejores atributos con MDE aplicando técnicas de árboles de decisión, sistemas de aprendizaje automático y técnicas basadas en casos, en densidad o distancia (Maya-Pérez, 2016).

Participar e inferir en el conocimiento generado por los estudiantes a partir de técnicas computacionales puede ser de gran utilidad, por ejemplo, puede ser una entrada significativa para otros tipos de análisis, puede ser útil para decidir cuándo avanzar un estudiante en el currículum o intervenir en otras vías, y además puede constituir información útil para los instructores en cuanto a su forma de proceder con los estudiantes (Corbett, 2014).

Los algoritmos de minería de datos requieren numerosos cálculos complejos y la colaboración entre personas con múltiples disciplinas y ubicaciones geográficas. La complejidad de los problemas modernos de minería de datos educacional, requiere un ambiente centrado en nuevas arquitecturas que propicien la utilización de entornos informáticos distribuidos y paralelos, para garantizar la escalabilidad e interactividad de los sistemas a medida que dichos datos crecen en cuanto a tamaño y complejidad (Dubitzky, 2008).

Los métodos tradicionales desarrollados, suponen que los datos residen mayoritariamente en memoria. Dicha suposición teniendo en cuenta las nuevas tecnologías, ya no es sostenible, debido a que la implementación de ideas de minería de datos en entornos informáticos distribuidos y paralelos de alto rendimiento se está convirtiendo en un método crucial para garantizar la escalabilidad y la interactividad del sistema (Zaki, 2000).

Los sistemas distribuidos constituyen un paso más en la evolución de los sistemas informáticos, concebidos desde el punto de vista de las necesidades que las aplicaciones plantean y las posibilidades que las tecnologías ofrecen. Los sistemas distribuidos ofrecen una gama de características, las cuales están basadas en realizar tareas de cómputo de alto rendimiento. Para

ello se apoyan en un conjunto de sistemas de cómputo interconectados de forma distribuida utilizando la red, que a su vez comparten un estado, dando una visión de sistema único (Lafuente, 2015).

Dentro del área de la minería de datos se han creado mecanismos para crear nuevos entornos distribuidos, tal es el caso de la Minería de Datos Distribuidos (MDD) o *Distributed Data Mining*. La MDD constituye un método descentralizado para el proceso de minería de datos, que es referida a la distribución de recursos a través de la red. Tiene como núcleo fundamental la escalabilidad y se centra en detalles importantes como la reutilización y la robustez, desarrollada en un entorno informático distribuido que tiene como objetivo intentar aprovechar al máximo los recursos disponibles tanto de hardware como de software. Existen dentro de esta rama diversas arquitecturas destinadas a la distribución y análisis de datos a través de algoritmos de minería de datos. Dichas arquitecturas describen entre sus principales características la aplicación de forma distribuida del procesamiento de datos, así como el control de los recursos disponibles y los factores humanos, prestando una especial atención hacia los datos distribuidos, la comunicación, los cálculos y los recursos para que se puedan utilizar de manera óptima.

Actualmente en la Universidad de las Ciencias Informáticas (UCI), específicamente en la Facultad 4, se encuentra el proyecto de investigación GITAE en la línea de Minería de Datos Educacionales Masivos, se estudia la aplicación de un modelo que colabore en el proceso de extracción de conocimiento valioso en datos educacionales. Dicho modelo se centra en que a partir de bases de datos educacionales heterogéneas, se apliquen técnicas de minado de datos, obteniéndose como resultados análisis predictivos de cuánto y cómo aprenden los estudiantes. El modelo cuenta con cuatro fases fundamentales. La primera de ellas es la gestión de conjunto de datos masivos, donde se aglomeran las bases de datos educacionales. Posteriormente, en la segunda fase, son aplicados los algoritmos de minería de datos a través de una plataforma para el minado de datos masivos, encargada de aglutinar los datos necesarios para la extracción de conocimiento para a través de los resultados del análisis predictivo mostrarlos en la fase de visualización de los resultados.

La plataforma para el minado de datos cumple un papel importante dentro del modelo desarrollado, la misma debe propiciar la conformación de un *dataset* (conjunto de datos) que contenga todas las bases de datos educacionales, y facilitar la comunicación del *dataset* con los algoritmos de minería de datos, lo que posibilita que puedan nutrirse y operar de forma distribuida para obtener a través de la plataforma el conocimiento estimado. Para el correcto funcionamiento de este sitio de fusión del *dataset* con los algoritmos es necesario tener en cuenta las 5V's del principio de la Minería de Datos, donde el volumen es asociado a la masividad de datos para el procesamiento, que se caracteriza por su gran variedad y heterogeneidad. La velocidad, necesaria para procesar dichos datos de la forma más eficiente posible, en cuanto al tiempo de respuesta con que son minados estos grandes conjuntos de datos.

Para ello, es necesario aplicar la minería de datos distribuida apoyada por las arquitecturas antes mencionadas, para garantizar la velocidad dentro de la plataforma. El estudio de estas arquitecturas posibilitó detectar las siguientes limitaciones y/o dificultades:

- ✓ Imposibilidad de los algoritmos de minería de datos destinados al análisis y procesamiento se nutran de todas las bases de datos con las que se desea trabajar.
- ✓ Imposibilidad de que los algoritmos de minería de datos operen de forma distribuida.

Para ello es necesario crear un entorno capaz de dar soporte a los algoritmos que serán aplicados, teniendo en cuenta a su vez el ambiente en el cual se debe aplicar dicho modelo, donde debe prevalecer la eficiencia en cuanto a la velocidad de procesamiento de los datos que son minados. Es necesario también tener en cuenta que la UCI centra sus bases en la utilización de software libre, para un desarrollo de la informática de forma sostenible y de alcance para todos.

Por lo anteriormente planteado se propone como **problema científico:**

¿Cómo contribuir al uso de Técnicas de Minería de Datos Distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos?

Definiendo como **objeto de estudio** la minería de datos distribuida y como **campo de acción** los entornos para el minado de datos distribuidos.

Se identifica como **objetivo de la investigación:**

Desarrollar un entorno distribuido para contribuir al uso de Técnicas de Minería de Datos Distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos.

Para dar cumplimiento al objetivo general planteado se proponen los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación donde se aborden los principales temas relativos a entornos distribuidos para el minado de datos masivos en entornos educacionales.
- ✓ Seleccionar las herramientas de hardware y software que soporten tecnológicamente el entorno.
- ✓ Desarrollar un entorno para el minado de datos en ambientes educacionales masivos.
- ✓ Validar la disponibilidad y la tolerancia a fallos del entorno para el minado de datos en ambientes educacionales masivos.

Como **hipótesis** de la investigación se platea:

Un entorno distribuido contribuye al uso de Técnicas de Minería de Datos Distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos.

Variables independientes: Entorno distribuido para el minado de datos en ambientes educacionales masivos.

Variables dependientes: Eficiencia en cuanto al tiempo de respuesta del procesamiento de datos.

Durante la investigación fueron empleados los métodos científicos que se citan a continuación (R. Sampieri, 2013):

Métodos teóricos:

- ✓ **Método analítico-sintético:** Utilizado para la descomposición del problema científico en elementos por separado y realizar una profundización en el estudio de cada uno de ellos, para luego sintetizarlos en la propuesta de solución.

- ✓ **Método inductivo-deductivo:** Utilizado para realizar el tránsito de los conocimientos generales a los particulares, en función de llegar a la conclusión del desarrollo de un entorno distribuido para el minado de datos en ambientes educacionales masivos.
- ✓ **Método histórico-lógico:** Empleado para el análisis de sistemas distribuidos, su utilización y principales características, además de la función de los mismos en trabajos de minería de datos, con el fin de un mejor entendimiento del objeto de estudio que se plantea en la investigación.
- ✓ **Método hipotético-deductivo:** Se hace uso del mismo para a partir de la hipótesis guiar la investigación. La observación y el análisis del fenómeno en cuestión hace posible la formulación de una hipótesis para guiar la investigación, la cual es comprobada en el proceso de validación.
- ✓ **Método sistémico:** Es empleado para garantizar que los elementos que son parte del entorno distribuido actúan como un todo, garantizando que los mismos trabajen de forma integral.
- ✓ **Experimento:** Sirvió como base para validar el cumplimiento de la hipótesis planeada en la investigación del entorno distribuido, desarrollado en un ambiente informático real para poder hacer constar la eficiencia de la propuesta de solución.

Métodos empíricos:

- ✓ **Análisis documental:** Realizado para ejecutar un estudio de los principales referentes teóricos de la investigación, a través de la consulta de libros y artículos científicos que garantizaran que el entorno distribuido desarrollado fuera de relevancia científica y aporte práctico.
- ✓ **Técnica Iadov:** Utilizado para el proceso de validar y obtener retroalimentación de los usuarios encuestados sobre el nivel de satisfacción de la propuesta de solución.
- ✓ **Criterio de expertos mediante el escalamiento de Likert:** Luego de realizada la encuesta a expertos sobre el tema en cuestión, fueron evaluados los elementos teóricos que fundamentan la presente investigación a partir de sus conocimientos y experiencias.
- ✓ **Encuesta:** Mediante su aplicación a especialistas de la rama informática sobre los procesos de minería de datos dentro del entorno distribuido desarrollado, fueron obtenidas mediciones de carácter cuantitativo de los elementos relacionados con los resultados de la investigación.

Se define como **aporte práctico**:

- ✓ El desarrollo de un entorno distribuido que permita procesar grandes volúmenes de datos educacionales masivos de forma eficiente.

La presente investigación está estructurada de forma general de la siguiente manera: resumen, índices, introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos.

CAPÍTULO 1:

Marco teórico referencial

Son abordados los elementos fundamentales que forman parte del objeto de estudio de la investigación, además de ser tratados los fundamentos teóricos para los procesos de minería de

datos. Se muestran los principales argumentos de utilización de entornos distribuidos y sus características, la minería de datos educacionales y la minería de datos en entornos distribuidos. Se realiza un análisis de las arquitecturas más comunes dedicadas al proceso de minado de datos distribuidos, para a partir de las mismas determinar los elementos a tener en cuenta para la elaboración de la propuesta de solución. Finalmente se efectúa un análisis de las herramientas novedosas y actualizadas que se utilizan a nivel mundial para el proceso de minado de datos y para la conformación del entorno distribuido.

CAPÍTULO 2:

Materiales y métodos

En el segundo capítulo se describe el flujo de cada proceso involucrado en la composición del entorno distribuido, haciendo énfasis en los principales subprocesos que interfieren en el proceso de minado de datos. Son explicados y aplicados los principales elementos con que deben contar un entorno distribuido para su conformación. Posteriormente, se realiza la composición del entorno con todos los elementos a tener en cuenta explicados anteriormente, además de describir cómo funciona cada uno de estos aspectos con las herramientas y el hardware seleccionado.

CAPÍTULO 3:

Análisis y discusión de resultados

Para el último capítulo, se procede al proceso de validación de la propuesta de solución a partir de los métodos científicos que fueron definidos anteriormente. Son validados los resultados obtenidos a través de la encuesta realizada para la aplicación de la técnica de Iadov, llevada a cabo con el objetivo de medir la satisfacción del usuario. Es utilizado, además, el método criterio de experto y finalmente para constatar el cumplimiento de la hipótesis científica se aplica el cuasi experimento.

Posteriormente de concluido el tercer capítulo son presentadas las conclusiones generales, son emitidas las recomendaciones, son listadas las referencias bibliográficas y son incluidos los anexos.

1 CAPÍTULO 1: FUNDAMENTOS TEÓRICOS Y METODOLÓGICOS

En el siguiente capítulo se abordan los principales temas relacionados con los entornos distribuidos, sus principales características y aplicación. Se realiza un estudio de conceptos tales como Minería de Datos (MD), Minería de Datos Educativos (MDE), Minería de Datos Distribuidos (MDD) y su utilización en ambientes educacionales masivos. Son analizadas las diferentes arquitecturas utilizadas a nivel mundial para el análisis de datos de forma distribuida, además de las principales herramientas y metodologías empleadas a nivel mundial en entornos distribuidos.

Sobre los entornos distribuidos se identificarán sus principales características, necesarias para su correcto funcionamiento. Se tratarán temas de la virtualización y la utilización de los mismos para el ahorro de hardware y la reducción de costos.

1.1 Minería de datos

La Minería de Datos (MD) o *Data Mining*, ha atraído una gran atención en la industria de la información y en la sociedad en general en los últimos años, debido a la amplia disponibilidad de enormes cantidades de datos y la inminente necesidad de convertir dichos datos en información y conocimiento útil. La información y el conocimiento adquirido puede ser utilizado para aplicaciones que van desde análisis de mercado, análisis de datos climatológicos, detección de fraude y retención de clientes, hasta control de producción y exploración científica. Dicho de manera simple, la minería de datos se refiere a extraer o "minar" el conocimiento de grandes cantidades de datos (Jiawei Han M. K., 2006) (Jain, 2015) (Jesús A. Castorena Peña, 2018) (Maria A. Murazzo, 2016).

Usualmente la minería de datos se atribuye como un sinónimo de otro término utilizado popularmente como *Knowledge Discovery from Data* (Descubrimiento de conocimiento de datos) o KDD por sus siglas en inglés. Alternativamente, otros ven la minería de datos como simplemente un paso esencial en el proceso de descubrimiento de conocimiento. El descubrimiento de conocimiento como un proceso consiste en una secuencia iterativa de los siguientes pasos (Jiawei Han M. K., 2006) (Ramageri, 2010) (Ocampo, 2017) (Lambodar Jena, 2015) (Skiena, 2017) (Aggarwal, 2015):

- ✓ Limpieza de datos (para eliminar ruido y datos inconsistentes).
- ✓ Integración de datos (donde se pueden combinar múltiples fuentes de datos).
- ✓ Selección de datos (donde los datos relevantes para el analista son recuperados de la base de datos).
- ✓ Transformación de datos (donde los datos se transfieren o se consolidan en una estructura apropiada para la minería al realizar operaciones de resumen o agregación).
- ✓ Minado de datos (un proceso esencial donde se aplican métodos inteligentes para extraer patrones de datos).
- ✓ Evaluación de patrones (para identificar los patrones realmente interesantes que representan el conocimiento basado en algunas medidas de interés).
- ✓ Presentación del conocimiento (donde las técnicas de visualización y representación del conocimiento se utilizan para presentar el conocimiento minado al usuario).

Como área de las Tecnologías de la Información y Comunicación (TIC), la MD integra numerosas técnicas de análisis de datos y extracción de modelos, capaz de obtener patrones, describir tendencias y regularidades, además de predecir comportamientos y explorar información computarizada, proveniente de fuentes heterogéneas y de grandes volúmenes de datos. Esta técnica permite a los individuos y a las organizaciones comprender y modelar de una manera más eficiente y precisa el contexto en que deben actuar y tomar decisiones inteligentes (Maya-Pérez, 2016) (Fúquene, 2017).

Para el desarrollo de la presente investigación a través de los conceptos antes estudiados, se define a la minería de datos como el área de las tecnologías de la información (TIC) dedicada a extraer conocimiento valioso de grandes volúmenes de datos, empleando novedosas técnicas de análisis de datos para predecir comportamientos y explorar información computarizada, proveniente de fuentes heterogéneas, capaz de obtener patrones, describir tendencias y regularidades.

1.2 Minería de datos educacional

El aumento de los recursos de *e-learning*, instrumentando los softwares educativos, el uso de Internet en la educación y el establecimiento de bases de datos estatales de la información del estudiante, ha creado grandes repositorios de datos. Uno de los mayores desafíos que las instituciones educativas es el crecimiento exponencial de los datos educativos y el uso de estos datos para mejorar la calidad de la toma de decisiones (Cristobal Romero, 2013) (Judy Kay, 2011).

En los últimos años, ha aumentado el interés en el uso de la minería de datos para investigar preguntas científicas dentro de la investigación educativa, un área de investigación denominada minería de datos educativos. La minería de datos educativos (MDE) se define como el área científica de investigación centrada en el desarrollo de métodos para hacer descubrimientos dentro de tipos de datos que provienen de entornos educativos. Esta área hace uso de métodos para comprender mejor a los estudiantes y la forma en la que estos aprenden, teniendo como apoyo el análisis del aprendizaje (*learning analytics*), el cual ha surgido como alternativa para el trabajo con datos educativos (Stefan Slater, 2017) (Baker R. S., 2011) (Kavitha & Raj, 2017, pág. 2017) (Ryan Baker, 2014) (Nabila Bousbia, 2014) (Ballesteros Román & Daniel Sánchez-Guzmán, 2014).

Otros autores atribuyen que la MDE analiza datos generados por cualquier tipo de sistema de información que apoye el aprendizaje o la educación (en escuelas, institutos, universidades y otras instituciones académicas o profesionales de aprendizaje que proporcionan formas y métodos tradicionales y modernos de enseñanza, así como el aprendizaje informal). Estos datos no están restringidos a interacciones de estudiantes individuales con un sistema educativo (por ejemplo, comportamiento de navegación, entrada en concursos y ejercicios interactivos) pero también puede incluir datos de estudiantes colaboradores (chat de texto), datos administrativos (escuela, distrito escolar, maestro), datos demográficos (sexo, edad, calificaciones escolares), afectividad del alumno (motivación, estados emocionales), y así sucesivamente (Cristobal Romero, 2013).

La MDE es un área interdisciplinaria que incluye, pero que no se limita a la recuperación de información, sistemas de recomendación, análisis de datos visuales, análisis de redes sociales (*Social Network Analysis*, SNA por sus siglas en inglés), psicopedagogía, psicología cognitiva,

psicometría, etc. De hecho, la MDE se puede modelar como la combinación de tres áreas principales como se muestra en la figura 1: informática, educación y estadística. La intersección de estas tres áreas también forma otras subáreas estrechamente relacionadas con la MDE, tales como la educación basada en computadora, el aprendizaje automático (*Machine Learning*) y el análisis de aprendizaje (*Learning Analytics*, LA) (Cristobal Romero, 2013) (Laura Calvet Liñán, 2015) (Nabila Bousbia, 2014).

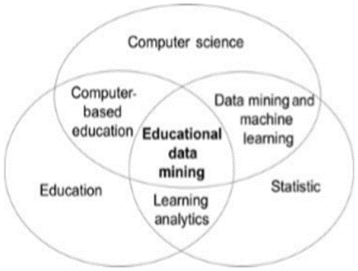


Figura 1: Principales áreas relacionadas con la Minería de Datos Educativos (Cristobal Romero, 2013) (Nabila Bousbia, 2014).

Teniendo en cuenta los conceptos antes citados, se asume para la presente investigación que la minería de datos educacional es un apéndice de la minería de datos, pero aplicado al contexto educativo. La misma funciona como un área interdisciplinaria que incluye subáreas, como la educación basada en computadora, el aprendizaje automático y el análisis de aprendizaje. Centra sus bases en el desarrollo de métodos para hacer descubrimientos dentro de tipos de datos que provienen de entornos educativos, y el uso de esos métodos para comprender mejor a los estudiantes y la forma en la que aprenden. Se concluye de esta manera, que la MDE analiza datos que son generados por cualquier tipo de sistema de información, necesaria para apoyar el aprendizaje o la educación.

1.3 Minería de datos distribuida

Como se planteó con anterioridad, la minería de datos es el proceso de extracción de información útil de conjuntos de datos utilizando técnicas de MD, concretamente emparejamiento de patrones, agrupamiento, asociación de reglas, regresión, etc. El crecimiento progresivo de la tecnología de la información ha allanado el camino para seguir explorando en los términos distribuido / colectivo de minado de datos, espacial y minería de datos geográficos, minería de datos temporales, minería de datos espacial-temporal, minería de datos multimedia y minería de datos fenomenales. La minería de datos distribuida (MDD) o *Distributed Data Mining* (por sus siglas en inglés), como también es conocida, tiene como objetivo extraer información útil de los datos ubicados en sitios heterogéneos. La informática distribuida comprende sitios distribuidos, que hospedan unidades informáticas en puntos heterogéneos individuales (Dillip Kumar Swain, 2017) (Chikhale, 2016) (Martínez, 2015). La MDD sigue una estrategia de minería descentralizada que difiere de la estrategia centralizada que hace que todo el sistema de trabajo sea escalable distribuyendo la carga de trabajo entre sitios heterogéneos. La figura 2 muestra lo anteriormente planteado (S.Urmela, 2017).

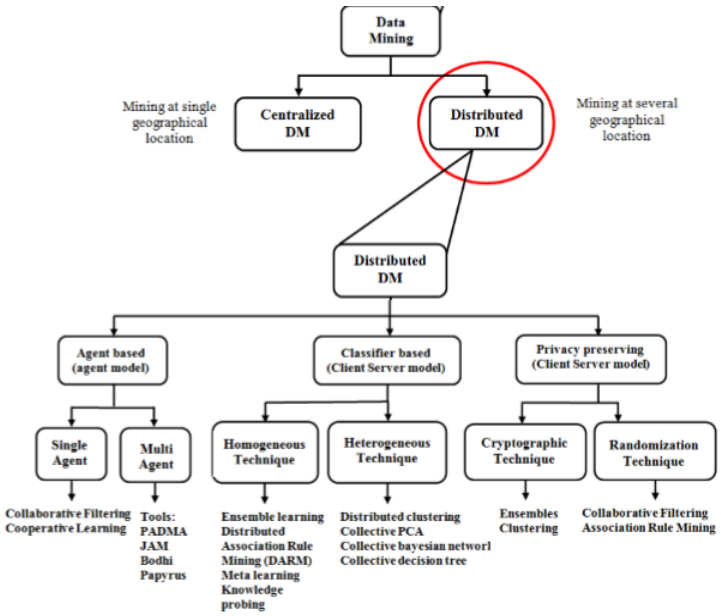


Figura 2: Clasificación de la MDD (S.Urmela, 2017).

Algunos autores afirman que la minería de datos es una rama del campo de minería de datos que ofrece un marco para extraer datos distribuidos prestando especial atención a los datos distribuidos y los recursos informáticos (D. Talia, 2006).

Otros afirman que la MDD es un sistema complejo, que se centra en la distribución de recursos a través de la red y en los procesos de minería de datos. Su núcleo está basado en la escalabilidad, donde la configuración del sistema puede verse alterada ocasionalmente, por lo tanto, el diseño de los sistemas de MDD se ocupa de los grandes detalles de reutilización, extensibilidad y robustez (Vuda Sreenivasa Rao S. V., 2010).

La MDD se ocupa fundamentalmente de la aplicación del procedimiento clásico de la minería de datos en un entorno informático distribuido que intenta aprovechar al máximo recursos disponibles ya sean redes de comunicación, unidades de cómputo y bases de datos. En la figura 3 se representa el enfoque de la MDD, donde la primera fase implica el análisis de la base de datos local en cada sitio distribuido. Posteriormente el conocimiento descubierto es transmitido generalmente a un sitio de fusión, donde la integración del modelo local distribuido es interpretada y los resultados se transmiten de vuelta a las bases de datos distribuidas para que todos los sitios se mantengan actualizados con el conocimiento global. En algunos enfoques en lugar de un sitio de fusión los modelos locales son transmitidos a todos los otros sitios, para que cada sitio pueda calcular el modelo global (Devi, 2014).

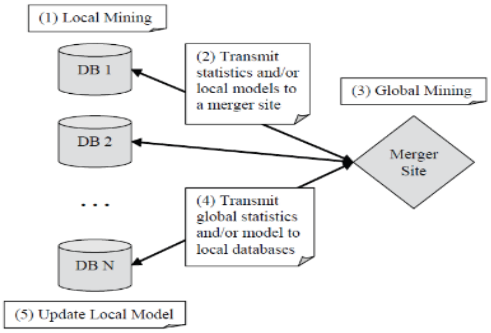


Figura 3: Enfoque de la Minería de Datos Distribuida (Devi, 2014).

Las técnicas paralelas de descubrimiento de conocimiento abordan este problema utilizando máquinas multicomputadora de alto rendimiento. La disponibilidad cada vez mayor de tales máquinas requiere un amplio desarrollo del análisis de datos, algoritmos que amplían su escala para analizar conjuntos de datos medidos en terabytes y petabytes en máquinas paralelas con cientos o miles de procesadores. Esta tecnología es particularmente adecuada para aplicaciones que generalmente tratan con una gran cantidad de datos (por ejemplo, datos de transacciones, datos científicos y datos de telecomunicaciones) que no se pueden ser analizadas en máquinas tradicionales en tiempos aceptables, por tanto puede ayudar esta tecnología en aumentar la eficiencia y reducir el costo de las redes de computadora al disminuir el tiempo de procesamiento de datos y optimizar recursos, además de distribuir las cargas de trabajo para permitir a los usuarios obtener resultados mucho más rápidos en grandes operaciones y a un costo menor (D. Talia, 2006). Existen diversos enfoques para la utilización de la minería de datos distribuida, como la utilización de métodos para la clasificación distribuida y la regresión. Otros enfoques distribuidos son adaptaciones de métodos de conjunto en un entorno informático distribuido, mientras que otros amplían los enfoques existentes para minimizar la comunicación y los costos que surgen con su aplicación. Diversos algoritmos son aplicados en esta área para extraer conocimiento, minimizar costos en cuanto a comunicación y sincronización de datos, reducir el tamaño de las transacciones y realizar modelos de clasificación y regresión sobre bases de datos heterogéneas (Devi, 2014).

1.4 Sistemas distribuidos

Un sistema distribuido es una colección de computadoras independientes que aparece a sus usuarios como un único sistema coherente. Esta definición posee varios aspectos importantes, el primero es que un sistema distribuido consta de componentes (es decir, computadoras) que son autónomos y un segundo aspecto es que los usuarios (entre personas o programas) piensan que están tratando con un solo sistema, lo que significa que de una forma u otra los componentes autónomos necesitan colaborar (Andrew S. Tanenbaum, 2007).

Otros, lo definen como un conjunto de computadores que se encuentran interconectados, compartiendo un estado y que ofrecen una visión de sistema único, mostrando con ello los recursos de manera homogénea, ocultando su distribución, donde el usuario y las aplicaciones no ven la red, sino un sistema indistinguible de uno centralizado. Como los componentes de un sistema distribuido

pueden ser heterogéneos, se requiere una capa de software llamada *middleware*, que se utiliza con el fin de proporcionar la visión de sistema único (Lafuente, 2015).

Sin embargo, otros autores definen que un sistema distribuido es aquel en el que los componentes ubicados en las computadoras en red se comunican y coordinan sus acciones solo mediante el envío de mensajes. Esta definición conduce a las siguientes características especialmente significativas de los sistemas distribuidos: concurrencia de componentes, falta de un reloj global y fallas independientes de los componentes (George Coulouris, 2012).

Una característica importante de los sistemas distribuidos es que las diferencias entre las diversas computadoras y las formas en que se comunican están mayormente ocultas a los usuarios. Otra característica importante es que los usuarios y las aplicaciones pueden interactuar con un sistema distribuido de manera consistente y uniforme, independientemente de dónde y cuándo tenga lugar la interacción (Andrew S. Tanenbaum, 2007).

Los sistemas distribuidos están experimentando un período de cambios importantes y esto se puede remontar a una serie de tendencias influyentes (George Coulouris, 2012):

- ✓ La aparición de una tecnología de red generalizada.
- ✓ La aparición de la informática ubicua junto con el deseo de apoyar la movilidad del usuario en los sistemas distribuidos.
- ✓ La creciente demanda de servicios multimedia.
- ✓ La vista de los sistemas distribuidos como una utilidad.

Para admitir computadoras y redes heterogéneas al tiempo que ofrece una vista de sistema único, los sistemas distribuidos a menudo se organizan por medio de un software, es decir, colocados lógicamente entre una capa de nivel superior compuesta por usuarios y aplicaciones, y una capa subyacente que consiste en operar sistemas y medios de comunicación básicos, como se muestra en la figura 8. Un sistema distribuido de esta manera es llamado en algunas ocasiones *middleware* (Andrew S. Tanenbaum, 2007).

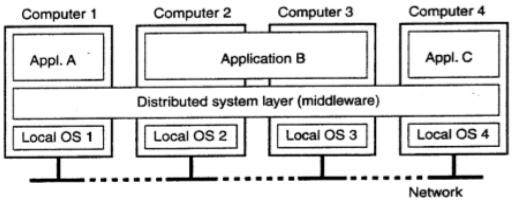


Figura 4: Sistema distribuido organizado como middleware. La capa de middleware se extiende a varias máquinas y ofrece a cada aplicación la misma interfaz (Andrew S. Tanenbaum, 2007).

La figura 4, muestra cuatro computadoras conectadas en red y tres aplicaciones, de las cuales la aplicación B se distribuye entre las computadoras 2 y 3. A cada aplicación se le ofrece la misma interfaz. El sistema distribuido proporciona los medios para que los componentes de una única aplicación distribuida se comuniquen entre sí, pero también para que se comuniquen diferentes aplicaciones. Al mismo tiempo, oculta lo mejor y lo más razonablemente posible, las diferencias en el hardware y los sistemas operativos de cada aplicación (Andrew S. Tanenbaum, 2007).

1.4.1 Propiedades de los sistemas distribuidos

La visión de sistema único está basada en el cumplimiento de propiedades que son descritas a continuación.

1.4.1.1 Transparencia

Un objetivo importante de un sistema distribuido es ocultar el hecho de que sus procesos y recursos están físicamente distribuidos en múltiples computadoras. Se dice que un sistema distribuido que puede presentarse a los usuarios y aplicaciones como si fuera un solo sistema informático es transparente (Andrew S. Tanenbaum, 2007).

Un sistema distribuido debe tener como principal objetivo proporcionar al usuario y a las aplicaciones una visión de los recursos del sistema como gestionados por una sola máquina virtual, donde la distribución física de los recursos es transparente. La transparencia es descrita a través de los siguientes aspectos (Lafuente, 2015) :

- ✓ Identificación: Los espacios de nombres de los recursos son independientes de la distribución de los recursos, de tal forma que una aplicación puede referirse a un recurso con nombre independiente del nodo en que se ejecute (Lafuente, 2015).
- ✓ Acceso: La transparencia de acceso se ocupa de ocultar las diferencias en la representación de datos y de que los usuarios puedan acceder a los recursos (Andrew S. Tanenbaum, 2007).
- ✓ Ubicación o Localización: La ubicación física de los recursos ni los usuarios ni las aplicaciones conocen en qué nodo reside el recurso accedido, o si el mismo es local o remoto, lo cual implica que los recursos tengan la posibilidad de migrar entre nodos sin que las aplicaciones se vean afectadas (Lafuente, 2015).
- ✓ Replicación: Se basa en que ni los usuarios ni las aplicaciones conocen cuántas unidades hay de cada recurso, ni si se añaden o eliminan copias del recurso (Lafuente, 2015). La transparencia de replicación trata de ocultar el hecho de que existen varias copias de un recurso (Andrew S. Tanenbaum, 2007).
- ✓ Paralelismo: Determina que una aplicación puede ejecutarse en paralelo, sin que la aplicación tenga que especificarlo, y sin que la misma tenga consecuencias sobre la ejecución, salvo por cuestiones que puedan presentarse en cuanto al rendimiento (Lafuente, 2015).
- ✓ Compartición: Referido a que un recurso que se encuentre compartido, intente ser accedido simultáneamente desde varias aplicaciones no tendrá efectos sobre la ejecución de dicha aplicación (Lafuente, 2015).
- ✓ Concurrencia: Un objetivo importante de los sistemas distribuidos es permitir el intercambio de recursos, donde en muchos casos, el intercambio de recursos se realiza de forma cooperativa, como en el caso de la comunicación (Andrew S. Tanenbaum, 2007).
- ✓ Fracaso: Hacer que una falla distribuida del sistema sea transparente, significa que el usuario no se da cuenta de que un recurso no funciona correctamente y que el sistema se recupera posteriormente de esa falla (Andrew S. Tanenbaum, 2007).

1.4.1.2 Escalabilidad

Una de las características fundamentales a tener en cuenta en los sistemas distribuidos es su modularidad, la cual le permite una gran flexibilidad y hace posible su escalabilidad, la que es definida como la capacidad del sistema para crecer sin aumentar su complejidad y que a su vez no disminuya su rendimiento. De los objetivos fundamentales del diseño de un sistema distribuido se encuentra la extensión de la escalabilidad a la integración de servicios (Lafuente, 2015).

La escalabilidad de un sistema se puede medir a lo largo de al menos tres dimensiones diferentes, primero, un sistema puede ser escalable con respecto a su tamaño, lo que significa que se puede agregar más usuarios y recursos al sistema. En segundo lugar, un sistema escalable geográficamente es uno en el que los usuarios y los recursos pueden estar muy separados. En tercer lugar, un sistema puede ser escalable desde el punto de vista administrativo, el cual puede ser fácil de administrar, incluso si abarca muchas organizaciones administrativas independientes. Desafortunadamente, un sistema que es escalable en una o más de estas dimensiones a menudo muestra alguna pérdida de rendimiento a medida que el sistema se amplía (Andrew S. Tanenbaum, 2007).

1.4.1.3 Fiabilidad y tolerancia a fallos

Una característica de los sistemas distribuidos que los distingue de los sistemas de una sola máquina es la noción de la ocurrencia de fallas parciales. Una falla parcial puede ocurrir cuando falla un componente en un sistema distribuido, dicha falla puede afectar el funcionamiento correcto de otros componentes, mientras que al mismo tiempo deja intactos otros componentes. Por el contrario, una falla en los sistemas no distribuidos suele ser total en el sentido de que afecta a todos los componentes y puede reducir fácilmente todo el sistema. Un objetivo importante en el diseño de sistemas distribuidos es construir el sistema de manera que pueda recuperarse automáticamente de fallas parciales sin afectar gravemente el rendimiento general. En particular, cada vez que ocurre una falla, el sistema distribuido debe continuar operando de manera aceptable mientras se realizan las reparaciones, es decir, debe tolerar las fallas y continuar operando (Andrew S. Tanenbaum, 2007).

La fiabilidad de un sistema se define como la capacidad para realizar correctamente y en todo momento las funciones para las que se ha diseñado el sistema distribuido, donde la misma se concreta a través de los siguientes aspectos (Lafuente, 2015):

- ✓ Disponibilidad: Se identifica por la fracción de tiempo que el sistema se encuentra operativo, su principal parámetro de medición es el tiempo medio entre fallos y además el tiempo en que tarda en repararse el sistema. Dicha disponibilidad se puede incrementar a través de dos cuestiones fundamentales: utilizando hardware de mayor calidad, y/o realizando un diseño basado en la replicación de componentes que de esta manera permita al sistema seguir operando en caso de que algunos de sus componentes fallen. Los sistemas distribuidos propician la replicación de recursos, la cual es necesaria para seguir incrementando la disponibilidad, ya que la probabilidad de fallo disminuye como una función exponencial de la propia replicación (Lafuente, 2015).

- ✓ Confiabilidad: Se refiere a la propiedad de que un sistema puede funcionar continuamente sin fallas. A diferencia de la disponibilidad, la fiabilidad se define en términos de un intervalo de tiempo en lugar de un instante en el tiempo. Un sistema altamente confiable es uno que probablemente continuará funcionando sin interrupción durante un período de tiempo relativamente largo (Andrew S. Tanenbaum, 2007).
- ✓ Tolerancia a fallos: A pesar de crear un sistema con una alta disponibilidad, un fallo en un determinado momento puede traer consigo graves consecuencias, aunque la replicación aumenta la disponibilidad, no garantiza por sí sola la continuidad del servicio de forma transparente. Es por ello que la tolerancia a fallos expresa la capacidad de que el sistema prosiga funcionando correctamente en caso de ocurrir algún fallo en uno de sus componentes, siendo a su vez completamente transparente al usuario y a la aplicación. Por tanto, un sistema con una buena tolerancia a fallos debe ser capaz de detectar el fallo y continuar el servicio, todo de forma transparente a la aplicación (Lafuente, 2015).

Si un sistema debe ser tolerante a fallas, lo mejor que puede ejecutar es tratar de ocultar la ocurrencia de fallas de otros procesos. La técnica clave para enmascarar fallas es usar redundancia, de la cual se describen tres tipos posibles (Andrew S. Tanenbaum, 2007):

- ✓ redundancia de información.
- ✓ redundancia de tiempo.
- ✓ redundancia física.

En la redundancia de información, se agregan bits adicionales para permitir la recuperación de bits ilegibles. Con la redundancia de tiempo, se realiza una acción y luego, si es necesario, se realiza nuevamente. La redundancia de tiempo es especialmente útil cuando las fallas son transitorias o intermitentes. La redundancia física, agrega equipos o procesos adicionales para que el sistema en su conjunto tolere la pérdida o el mal funcionamiento de algunos componentes. Este tipo de redundancia se puede hacer tanto en hardware como en software, por ejemplo, se pueden agregar procesos adicionales al sistema, de modo que si un pequeño número falla, el sistema aún puede funcionar correctamente. En otras palabras, al replicar procesos, se puede lograr un alto grado de tolerancia a fallos (Andrew S. Tanenbaum, 2007).

1.4.1.4 Consistencia

Como se ha tratado anteriormente la distribución de recursos contribuye a brindar importantes beneficios como es el incremento del rendimiento a través del paralelismo y el acceso a copias locales de recursos. Por otro lado, la replicación hace posible el aumento de la disponibilidad, constituyendo la base para proporcionar tolerancia a fallos en un sistema distribuido. No obstante, distribuir recursos trae consigo algunos problemas, como es el hecho de la red de interconexión, la seguridad del sistema es más vulnerable ante accesos no permitidos y el problema de mayor complejidad es el de la gestión del estado global para evitar situaciones de inconsistencia entre componentes del sistema, el cual es un aspecto fundamental en el diseño del sistema distribuido. Esencialmente el problema se ubica en la necesidad de mantener un estado global consistente en un sistema con varios componentes, cada uno de los cuales posee su estado local (Lafuente, 2015).

1.4.2 Clase de sistemas distribuidos

Una clase importante de sistemas distribuidos es la que se utiliza para las tareas informáticas de alto rendimiento. En términos generales, se distinguen en dos subgrupos, la informática de *cluster*, donde el hardware subyacente consiste en una colección de estaciones de trabajo o PC similares, estrechamente conectadas mediante una red de área local de alta velocidad y cada nodo ejecuta el mismo sistema operativo, y la computación Grid. Este subgrupo consiste en sistemas distribuidos que a menudo se construyen como una federación de sistemas informáticos, donde cada sistema puede pertenecer a un dominio administrativo desigual y puede ser muy diferente cuando se trata de hardware, software y tecnología de red desplegada (Andrew S. Tanenbaum, 2007).

1.4.2.1 Clustered Data Mining

Un enfoque de concebir un entorno de minería de datos distribuido es a través de *clusters* o comúnmente denominado *Clustered Data Mining*, el cual tiene como objetivo descubrir conocimiento de diferentes fuentes de datos distribuidas en múltiples nodos y combinarlo para construir un modelo de minería de datos global (Mafruz Zaman Ashrafi, 2002).

Los *clusters*, construidos con componentes de hardware de fábrica, así como el software gratuito o de uso común, juegan un papel importante en la redefinición del concepto de supercomputación. En consecuencia, han surgido como plataformas paralelas y distribuidas para computación de alto rendimiento y de alta disponibilidad (Yeo C.S., 2006).

Los sistemas de computación en *cluster* se hicieron populares cuando mejoró la relación precio/rendimiento de las computadoras personales y las estaciones de trabajo. Prácticamente en todos los casos, la computación en *cluster* se usa para programación paralela en la que se ejecuta un único programa (de cálculo intensivo) en paralelo en máquinas múltiples (Andrew S. Tanenbaum, 2007).

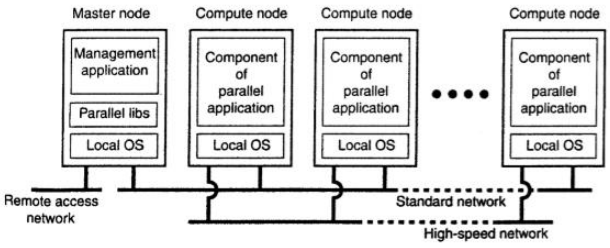


Figura 5: Ejemplo de sistema de computación en clúster (Andrew S. Tanenbaum, 2007).

Como se muestra en la figura 5, cada *cluster* consiste en una colección de nodos de cálculo que se controlan, y que son accedidos por medio de un solo nodo *master*. El *master* normalmente maneja la asignación de nodos a un programa paralelo en particular, mantiene una cola de proceso por lotes de trabajos enviados y proporciona una interfaz para los usuarios del sistema. El *master* realmente ejecuta el *middleware* necesario para la ejecución de programas y la administración del *cluster*, mientras que los nodos de cómputo a menudo no necesitan nada más que un sistema operativo estándar (Andrew S. Tanenbaum, 2007).

La tendencia de la computación paralela, es alejarse de las plataformas de supercomputación tradicionales especializadas a sistemas más económicos y de uso general, que consisten en

componentes débilmente acoplados creados a partir de computadoras personales, multiprocesadores o estaciones de trabajo. Esta ventaja incluye los bajos costos de acceso para acceder al desempeño de nivel de supercomputación, la capacidad de rastrear tecnologías, el sistema de actualización incremental, las plataformas de desarrollo de código abierto y la independencia del proveedor. Hoy en día, los *clusters* se utilizan ampliamente para la investigación y desarrollo de aplicaciones que exigen cálculos de alto rendimiento. Además, los *clusters* abarcan puntos fuertes como la alta disponibilidad y la escalabilidad (Yeo C.S., 2006).

La arquitectura típica de un *cluster* se muestra en la figura 6, donde los componentes claves del mismo incluyen múltiples computadoras independientes (PC, *Workstations* o SMP), sistemas operativos, interconexiones de alto rendimiento, *middleware*, entornos de programación paralelos y aplicaciones (Yeo C.S., 2006).

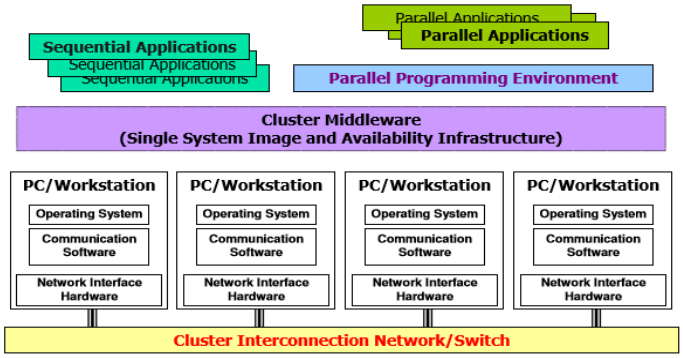


Figura 6: Arquitectura de un cluster (Yeo C.S., 2006).

Las computadoras *clusters* se pueden clasificar en tres clases, *cluster* de productos, de propiedad y de constelaciones. El *cluster* de productos es conocido como *cluster* de clase *Beowulf* y se puede construir principalmente a partir de componentes de hardware básicos, como computadoras multiprocesadores simétricos de bajo costo (SMP) y Ethernet, junto con sistemas operativos libres tales como Linux. Los *clusters* de propiedad están diseñado a medida, donde uno o más componentes se utilizan para ofrecer características de sistemas superiores. Los *clusters* de constelaciones es un conjunto de SMP que enfatiza el paralelismo de cada nodo SMP, de modo que incorpora más procesadores por nodo SMP que la cantidad de nodos existentes. Existen varios estudios de aplicación de arquitecturas basadas en *clusters* desarrolladas para diversas áreas, ya sean para realizar descubrimientos de conocimiento paralelo a partir de grandes fuentes de datos, utilizando algoritmos de minería de datos para combinar reglas de minería generadas a partir de diferentes nodos (Mafruz Zaman Ashrafi, 2002).

1.4.2.2 Data Mining Grid

El concepto de minería de datos distribuida basada en Grid (malla de computadoras), en analogía Data Grid o computación basada en Grid (*Computational Grid*) se define como una malla de minería de datos (*Data Mining Grid*), la cual busca una solución de compromiso entre la centralización de datos y el procesamiento distribuido de datos para maximizar la eficacia y la eficiencia de todo el proceso. Una *Data Mining Grid* debe proporcionar un medio para explotar los recursos de hardware disponibles (memoria primaria / secundaria, procesadores) con el fin de manejar los volúmenes de

datos y los requisitos de procesamiento de las aplicaciones modernas de minería de datos. La llamada red Grid es una infraestructura de computación distribuida que facilita la compartición coordinada de recursos informáticos dentro de las organizaciones y entre los sitios geográficamente dispersos. Esencialmente, un entorno de minería de datos habilitado para Grid consta de una plataforma de computación de alto rendimiento descentralizada donde las tareas de minería de datos y algoritmos pueden aplicarse en datos distribuidos (Dubitzky, 2008).

Un entorno basado en Grid provee instalaciones informáticas de alto rendimiento y acceso transparente a ellos a pesar de su ubicación remota, diferentes dominios administrativos, además de hardware y software con características heterogéneas. Se distingue principalmente de los mecanismos convencionales de informática distribuida por su enfoque de intercambio de recursos a gran escala, aplicaciones innovadoras y orientación de alto rendimiento (Devi, 2014).

La computación Grid se centra en la integración de los recursos existentes, dígase hardware, sistemas operativos, administración de recursos locales e infraestructura de seguridad. La interoperabilidad y la seguridad son las principales preocupaciones de esta infraestructura (Shruti N. Pardeshi, 2013).

La computación Grid se identifica como una herramienta necesaria para el almacenamiento y análisis de volúmenes elevados de información que diversas disciplinas generan. Es por ello que el conocimiento de la arquitectura Grid se hace necesario en los entornos académicos, con el propósito de dar a conocer cuáles son las experiencias que los estudiantes pueden adquirir en el entorno Grid, entre otros factores de interés, pero los costos del hardware necesario para implementar una arquitectura de este tipo desbordan los presupuestos de las entidades educacionales públicas (E. E. Millán-Rojas, 2016).

La diferencia existente entre el cómputo en *cluster* y la Grid está dada en que la computación en *cluster* presenta un alto grado de homogeneidad. En la mayoría de los casos, las computadoras de un *cluster* son prácticamente iguales, todas tienen el mismo sistema operativo y están conectadas a través de la misma red. Por el contrario, los sistemas de computación Grid tienen un alto grado de heterogeneidad, no se hacen suposiciones sobre hardware, sistemas operativos, redes, dominios administrativos, políticas de seguridad, entre otros factores (Andrew S. Tanenbaum, 2007).

Teniendo en cuenta las características de la minería de datos distribuida basada en Grid y el *clusterd data mining*, así como el ambiente de trabajo controlado que necesita el entorno distribuido para el minado de datos en ambientes educacionales masivos, es seleccionado este último para su aplicación. El mismo será aplicado para garantizar la seguridad e integridad de los datos que son minados.

1.4.3 Entornos distribuidos para el minado de datos

En la actualidad, muchas organizaciones, empresas y centros científicos producen y administran grandes cantidades de datos e información compleja. Los datos climáticos, los datos astronómicos y los datos de transacciones de compañías, son solo algunos ejemplos de cantidades masivas de repositorios de datos digitales que hoy deben almacenarse y analizarse para encontrar conocimiento útil en ellos. Este patrimonio de datos e información, se puede aprovechar de manera

efectiva si se usa como fuente para producir el conocimiento necesario para respaldar la toma de decisiones (D. Talia, 2006).

Los problemas científicos fundamentales que se exploran actualmente generan datos complejos, los cuales requieren simulaciones más realistas de los procesos y exigir visualizaciones más grandes y complejas de los resultados. Estos problemas a menudo requieren numerosos cálculos complejos y la colaboración entre personas con múltiples disciplinas y ubicaciones geográficas. La complejidad de los problemas modernos de minería de datos es un desafío para investigadores y desarrolladores, por lo cual la magnitud de dichos problemas requiere nuevas arquitecturas informáticas, ya que los sistemas convencionales ya no pueden hacerle frente (Dubitzky, 2008).

Los métodos tradicionales suelen suponer que los datos residen en la memoria, dicha suposición basada en las nuevas tecnologías ya no es sostenible. La implementación de ideas de minería de datos en entornos informáticos distribuidos y paralelos de alto rendimiento, se está convirtiendo en crucial para garantizar la escalabilidad e interactividad del sistema a medida que los datos continúan creciendo inexorablemente en tamaño y complejidad (Zaki, 2000).

En un entorno distribuido típico, el análisis de datos distribuidos es un problema no trivial debido a muchas limitaciones tales como el ancho de banda limitado (por ejemplo, las redes inalámbricas), datos sensibles a la privacidad, nodos de cómputo distribuidos, entre otros muchos casos. La Minería de Datos Distribuida es empleada en estos casos cuando los conjuntos de datos son de gran escala, debido a que la velocidad de la tarea de minería de datos es crucial (D. Talia, 2006).

Enmarcar una metodología para MDD es un desafío no solo por el entorno distribuido, sino también por su uso compartido de recursos eficiente y especificaciones de complejidad computacional minimizadas. La MDD se compone principalmente de dos variaciones: datos distribuidos y computación distribuida. En el primer método, los datos se distribuyen entre sitios heterogéneos a nivel local y los cómputos se alojan a nivel global. En el segundo método, el cálculo se distribuye entre sitios heterogéneos a nivel local y los datos se alojan a nivel global. La figura 7 que se muestra a continuación explica la arquitectura de trabajo de MDD, donde la base de datos de sitios heterogéneos alberga información útil y desconocida, y los algoritmos de MDD se aplicarán sobre los datos en sitios heterogéneos como modelo local y, finalmente, los resultados computados de MD se aglomerarán para formar un modelo global (S.Urmela, 2017) (Vuda Sreenivasa Rao S. V., 2010).

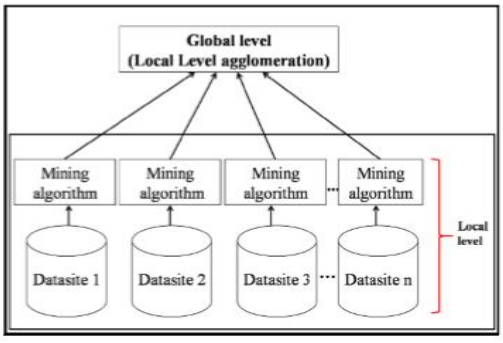


Figura 7: Arquitectura de trabajo con Minería de Datos Distribuida (S.Urmela, 2017).

Varios investigadores analizaron la complejidad involucrada en la metodología de enmarcado para MDD de dos maneras: analizando el uso efectivo y eficiente de recursos computacionales en sitios de datos individuales distribuidos, realizando descubrimiento de conocimiento en un sitio de datos distribuidos individuales (nivel local) y agregando conocimiento descubierto en el nivel global. Los resultados arrojaron que existen ciertos problemas en el desarrollo de algoritmos MDD, concretamente formulando algoritmos de MDD adecuados para conjuntos de datos heterogéneos, minimizando la complejidad computacional y del espacio, mejorando la privacidad de los datos en sitios distribuidos y manteniendo la autonomía de los conjuntos de datos locales. Además, todos estos problemas están interrelacionados entre sí (S.Urmela, 2017).

Otros autores, han desarrollado una arquitectura para MDD donde el procesamiento se lleva a cabo localmente en sitios de datos individuales, donde finalmente los datos se acumulan para formar un nivel global (S.Urmela, 2017). Por su parte, otros autores han presentado una arquitectura para la MDD donde el conocimiento adquirido de los sitios de datos distribuidos locales se acumula a nivel global formando un sitio de fusión (Tsoumakas G, 2008).

1.4.4 Arquitecturas para la minería de datos distribuida

La arquitectura de datos distribuida define tres variantes fundamentales para su utilización. La primera variante está dada en que cada procesador o nodo distribuido dispone de un componente de minería encargado de minar los datos en una base de datos local, obteniéndose como resultado un modelo de minería de datos parcial en cada nodo. Finalmente estos modelos parciales obtenidos se combinan para obtener un modelo de minería de datos global (Rodríguez Z. E., 2015) (Thampi, 2010). Esta primera variante es mostrada en la figura 8.

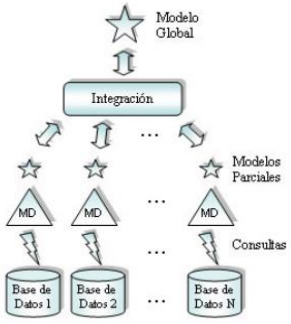


Figura 8: Arquitectura MDD con componentes locales (Rodríguez Z. E., 2015).

La segunda variante realiza consultas en cada base de datos distribuida de forma independiente según el conjunto de datos a analizar, posteriormente dichas consultas integradas conforman la vista de datos sobre la que operan los algoritmos de minería de datos. A continuación en la figura 9 es mostrada la arquitectura antes descrita (Rodríguez Z. E., 2015).

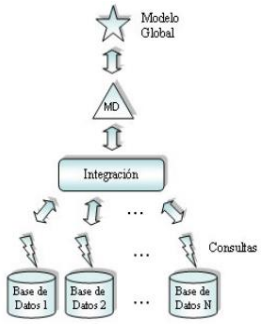


Figura 9: Arquitectura MDD con componentes globales e integración de resultados (Rodríguez Z. E., 2015).

La tercera variante que se observa en la figura 10, integra todas las bases de datos distribuidas y las consultas se realizan sobre esta vista integrada de datos y no en cada base de datos distribuida de manera independiente (Rodríguez Z. E., 2015).

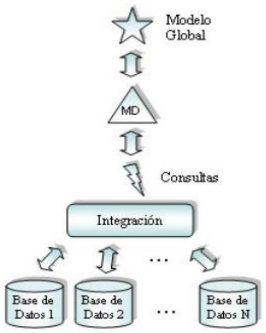


Figura 10: Arquitectura MDD con componentes globales e integración de locales (Rodríguez Z. E., 2015).

Después de analizada cada una de las arquitecturas, se puede identificar que en la figura 8 resulta de inconveniente la imposibilidad de que no todos los algoritmos tengan conocimiento de todas las bases de datos, es decir que los algoritmos no se nutren de las mismas bases de datos y pueden expedir resultados totalmente diferentes.

En el caso de las figuras 9 y 10 se evidencia que no se obtienen modelos de minería de datos parciales, sino únicamente un modelo de minería de datos global. Dichas arquitecturas muestran como inconveniente la imposibilidad de que los algoritmos de minería de datos operen de forma distribuida, realizando el proceso de análisis de datos mediante los algoritmos únicamente de forma centralizada.

1.5 Virtualización

La virtualización consiste en la extracción del software de una computadora encapsulado en un término llamado máquina virtual o *virtual machine* (VM), el cual es ejecutado en una máquina física ajena a la virtual. Las ventajas de la virtualización son disímiles, pero la más evidente es el ahorro de hardware, lo que conlleva a una reducción de costos en mantenimiento y administración de la Intranet, además de reducciones en el consumo eléctrico. La virtualización también ofrece ventajas en cuanto a la seguridad y movilidad de los sistemas. La realización de copias de seguridad se

presenta como un servicio desarrollado por cualquier software de virtualización, en el que al permitir copias de la máquina virtual completa facilita su traslado a nuevas ubicaciones. La virtualización es el término utilizado para la abstracción de los recursos de una computadora y su puesta en funcionamiento como máquina virtual en otra máquina física. Las máquinas virtuales utilizan el concepto de *hypervisor* para monitorear diferentes sistemas operativos invitados. El sistema operativo invitado en la máquina virtual se ejecuta con un *kernel* separado, por lo tanto, con el *kernel* del sistema operativo *host*, cada sistema operativo invitado tendrá un *kernel* separado, lo que se traduce en un aumento de la sobrecarga y un mayor tiempo de inicio. El *hypervisor* también proporciona aislamiento entre los diferentes sistemas operativos de los huéspedes, pero debido al *kernel* separado, el aislamiento del proceso resulta muy costoso. Las máquinas virtuales tradicionales carecen de la minimización de la sobrecarga de recursos, más tiempo de inicio y no pueden tolerar fallas (Shrikant S. Raut, 2016) (Diego Martín, 2011). La figura 11 muestra la composición de una máquina virtual tradicional

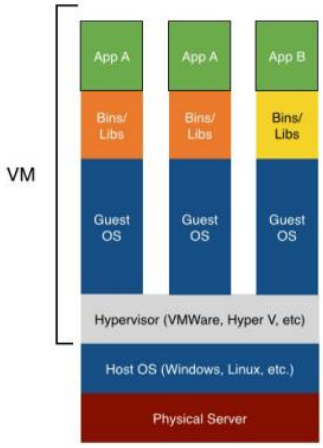


Figura 11: Máquina virtual tradicional (Shrikant S. Raut, 2016).

1.5.1 Docker

Docker es una plataforma abierta de computación en la nube para el *hosting* de aplicaciones de forma distribuida. Dicha plataforma está basada en una tecnología conocida como "*Linux Containers*", la cual es conocida como "Virtualización Ligera". Dicha herramienta ayuda a resolver problemas comunes como instalar, eliminar, actualizar, distribuir y administrar software. Está basado sobre Linux, por lo cual se define como una herramienta de código abierto. Docker provee una terminología llamada abstracción, la cual permite trabajar con técnicas complicadas en términos simplificados, en el caso de esta herramienta en particular, en lugar de enfocarse en todas las complejidades y detalles asociados con la instalación de una aplicación, todo lo que se necesita es considerar que software se desea instalar (Nickoloff, 2016) (Diliu, 2014) (Javier Rey, 2015) (Nikyle Nguyen, 2017) (Turnbull, 2018).

1.5.1.1 Contenedores en Docker

Un contenedor en Docker solo contiene el código de la aplicación y sus dependencias para su ejecución y es manejado por lo que se llama Docker Engine. Dicho contenedor es ejecutado como un proceso aislado en espacio de usuario en el sistema operativo, compartiendo el *kernel* con otros

contenedores. Los contenedores aíslan el software de su entorno, son definidos como una abstracción de la capa de aplicaciones que combina código y dependencias. Se pueden ejecutar varios contenedores en la misma máquina y compartir a su vez el núcleo del sistema operativo con otros contenedores, donde cada uno de los cuales se ejecuta como procesos aislados en el espacio de usuario. La figura 12 muestra la ejecución de Docker a partir de dos programas en espacio de usuario. El primero es el *daemon* Docker, mientras que el segundo es el Docker CLI, el cual se describe como el programa en que los usuarios interactúan con Docker. También se muestran tres contenedores en funcionamiento, los cuales se ejecutan como procesos secundarios del *daemon* de Docker, envuelto en un contenedor y la delegación de procesos es ejecutada en su propio subespacio de memoria del espacio de usuario (Turnbull, 2018) (Nickoloff, 2016).

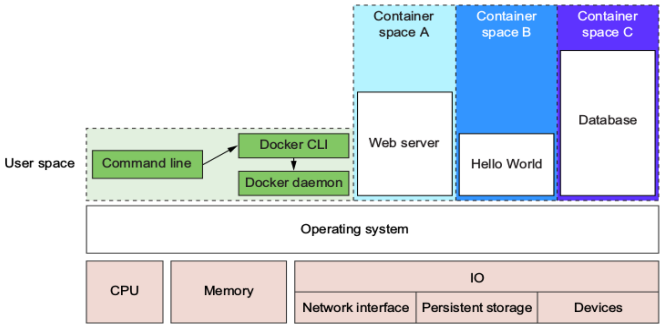


Figura 12: Ejecución de tres contenedores en un sistema operativo Linux (Nickoloff, 2016).

Docker puede ejecutar, copiar y distribuir contenedores con facilidad, esta tecnología completa la metáfora del contenedor tradicional al incluir una forma de empaquetar y distribuir el software, donde al componente que llena la función del contenedor se le llama imagen. Una imagen en un contenedor es un paquete ligero, autónomo y ejecutable de una pieza de software que incluye todo lo necesario para ser ejecutado: código, tiempo de ejecución, herramientas del sistema y configuraciones, disponibles para aplicaciones basadas en Linux y Windows, donde dicho software en el contenedor siempre funcionará igual independientemente del entorno en el cual se desarrolle. Una imagen en Docker es una instantánea agrupada de todos los archivos que debería estar disponible para un programa que se ejecuta dentro de un contenedor. Se puede crear tantos contenedores desde una imagen como sean necesarios, al crearlos los contenedores que se iniciaron desde la misma imagen no comparten los cambios en su sistema de archivos (Nickoloff, 2016) (Turnbull, 2018).

1.5.1.2 Seguridad en Docker

Desde el punto de vista de la seguridad en Docker se puede afirmar que los contenedores protegen al software que se ejecuta dentro de un contenedor. Existen diferentes vías por las cuales pueden presentarse riesgos en cuanto a la seguridad del sistema (Nickoloff, 2016):

- ✓ Un programa podría haber sido escrito específicamente por un atacante.
- ✓ Los desarrolladores podrían escribir programas con errores dañinos.
- ✓ Un programa accidentalmente posee errores dañinos a través de errores en su manejo de entrada.

El uso de un software pone en riesgo la seguridad de las estaciones de trabajo, es por ello que proteger la seguridad de la misma es prudente aplicando mitigaciones de riesgo prácticas (Nickoloff, 2016).

Como estrategias para mitigar los riesgos que puedan presentarse, cualquier software dentro de un contenedor solo puede acceder a elementos que están dentro de los elementos también. Pueden existir excepciones a esta regla, pero solamente cuando sea especificado por el propio usuario. Los contenedores limitan el alcance del impacto que un programa puede tener en otros programas en ejecución, los datos a los que puede acceder y los recursos del sistema. La figura 13 ilustra la diferencia entre ejecutar el software dentro y fuera de un contenedor (Nickoloff, 2016).

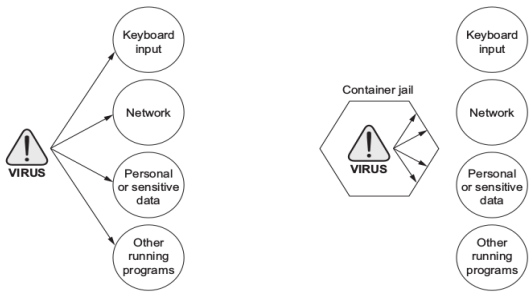


Figura 13: A la izquierda: Un programa malicioso con acceso directo a recursos confidenciales. A la derecha: Un programa malicioso dentro de un contenedor (Nickoloff, 2016).

Los contenedores no son una solución total para problemas de seguridad, pero se pueden utilizar para evitar muchos tipos de ataques. Otra cuestión importante a tener en cuenta es la no utilización de software de fuentes que no sean confiables, específicamente si los mismos requieren privilegios administrativos, lo cual significa que no es recomendable ejecutar a ciegas contenedores proporcionados por entornos compartidos.

1.5.1.3 Docker Swarm

Swarm es una herramienta simple que controla un grupo de *hosts* de Docker y lo expone como un solo *host* virtual. Dicha herramienta utiliza la API Docker estándar como su interfaz, lo que significa que cualquier herramienta que diga Docker puede controlar de forma transparente el Swarm. El Swarm Mode existe de forma nativa para el motor de Docker y se define como la capa que existe entre el sistema operativo y el contenedor de la imagen (Nickoloff, 2016) (Fabrizio Soppelsa, 2016).

El *clustering* es una característica importante para la tecnología de contenedores ya que crea un grupo cooperativo de sistemas que puedan proporcionar redundancia. Esto permite que el Docker Swarm conmute por error, si uno o más nodos experimentan un *outage* (corte de luz). Un *cluster* Swarm está formado por dos tipos de máquinas, una que ejecuta el Swarm en modo *master* (Swarm Manager) y otra máquina que ejecuta en modo *worked* (nodo trabajador o Swarm Agent) (Nickoloff, 2016).

Un *cluster* en Docker Swarm proporciona a los administradores y desarrolladores la posibilidad de añadir y sustraer iteraciones de contenedores como demandan los cambios computacionales. El gestor Swarm permite a un usuario crear un gestor principal y múltiples instancias de réplica en caso de que la instancia principal falle. En el motor de Docker Swarm el usuario puede desplegar el administrador y los nodos trabajadores en tiempo de ejecución. Los nodos *masters* y los nodos

trabajadores son como cualquier otra máquina de Docker, los cuales no requieren instalación especial y los mismos son ejecutados en los contenedores de Docker. La figura 16 ilustra una máquina *master* en modo Swarm y una máquina virtual típica corriendo el motor de Docker (Nickoloff, 2016).

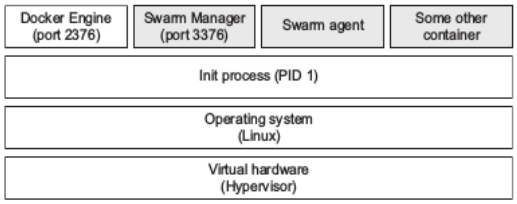


Figura 14: Docker Swarm ejecutado en un nodo master y otro programa ejecutado en un contenedor (Nickoloff, 2016).

Docker Swarm compone una API donde los clientes de Swarm pueden utilizarla para controlar o inspeccionar un nodo trabajador. Dicha API es una extensión de Docker Remote API, lo que significa que cualquier cliente Docker pude conectarse directamente al Swarm y tratar a un nodo trabajador como si fuera una sola máquina (Nickoloff, 2016).

Los nodos *managers* son las únicas máquinas en un Swarm que pueden ejecutar sus comandos, o autorizar a otras máquinas a unirse al modo Swarm como nodos trabajadores, los cuales se utilizan para proporcionar capacidad y no poseen autoridad para desempeñarse como administradores de las demás máquinas (Nickoloff, 2016).

Algunas de las características más destacadas se describen a continuación (Nickoloff, 2016):

- ✓ Administración de *clusters* integrada con Docker Engine: Se utiliza para crear un Swarm de Docker Engine, donde se puede implementar varios servicios de aplicaciones, el cual no necesita ningún otro software de orquestación adicional para crear o administrar un Swarm.
- ✓ Diseño descentralizado: En lugar de manejar la diferenciación entre los roles de los nodos en el momento de despliegue, el motor de Docker maneja cualquier especificación en tiempo de ejecución. Se pueden construir ambos tipos de nodos *managers* y trabajadores, lo que significa que se puede construir un Swarm completo a partir de una sola imagen.
- ✓ Escalabilidad: Para cada servicio se puede declarar el número de tareas que se desea ejecutar, el Swarm *manager* se adapta automáticamente al agregar o eliminar tareas para mantener el estado deseado.
- ✓ Conciliación de estado deseado: El nodo *master* supervisa constantemente el estado del *cluster* y reconcilia las diferencias entre el estado real y el estado deseado expresado.
- ✓ Equilibrio de carga: Puede exponer los puertos para servicios a un equilibrador de carga externo, internamente el Swarm permite especificar como distribuir contenedores de servicios entre nodos.

- ✓ Seguridad de forma predeterminada: Cada nodo del Swarm aplica autenticación y cifrado mutuo TLS para asegurar las comunicaciones entre sí y entre los demás nodos. Tiene la opción se usar certificados raíz autofirmados desde una raíz personalizada.

1.5.2 Docker vs Máquinas virtuales

Un contenedor, como se abordó anteriormente se ejecuta de forma nativa en Linux y comparte el *kernel* de la máquina *host* con otros contenedores, el cual se ejecuta como un proceso discreto, no teniendo más memoria que cualquier otro ejecutable, por lo que se considera ligero. Cada contenedor pude *booter* casi mil veces más rápido que una máquina virtual, y usar menos espacio y menos RAM. Docker provee un modo de ejecutar casi cualquier aplicación de forma segura aislada en un contenedor, el aislamiento y la seguridad proporcionada, lo que permite ejecutar múltiples contenedores en un *host*. La naturaleza ligera de los contenedores, que son ejecutados sin la penalización extra de un *hypervisor*, lo que significa una mejor utilización del hardware disponible. En la figura 15 se muestra la diferencia que existe entre el enfoque de una máquina virtual (VM) y un contenedor en Docker (Nickoloff, 2016).

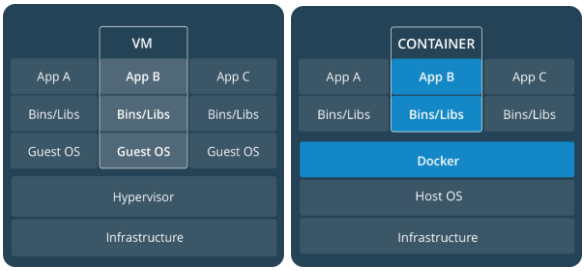


Figura 15: Comparación entre las arquitecturas de una Máquina Virtual (VM) y un contenedor de Docker (Nickoloff, 2016) (Shrikant S. Raut, 2016).

Los contenedores son una abstracción en la capa de aplicaciones que combina código y dependencias. Se pueden ejecutar varios contenedores en la misma máquina y compartir el núcleo del sistema operativo con otros contenedores, cada uno de los cuales se ejecuta como procesos aislados en el espacio de usuario. Los mismos ocupan menos espacio que las máquinas virtuales (las imágenes de los contenedores suelen tener decenas de MB de tamaño) y comienzan casi al instante de ser ejecutados (Nickoloff, 2016).

Las máquinas virtuales (VM), son una abstracción del hardware físico que convierte un servidor en muchos servidores. El *hypervisor* permite que múltiples máquinas virtuales se ejecuten en una sola máquina. Cada VM incluye una copia completa de un sistema operativo, una o más aplicaciones y bibliotecas necesarias, ocupando decenas de GB, las cuales también pueden tardar en arrancar (Nickoloff, 2016).

Los contenedores y las máquinas virtuales tienen beneficios similares de aislamiento y asignación de recursos, pero funcionan de manera diferente debido a que los contenedores virtualizan el sistema operativo en lugar del hardware, y son más portátiles y eficientes (Nickoloff, 2016).

Con respecto al almacenamiento en disco, una máquina virtual puede ocupar varios gigas ya que tiene que contener el sistema operativo completo. Sin embargo los contenedores en Docker solo

contienen aquello que las diferencia del sistema operativo en las que se ejecutan, por ejemplo un sistema operativo Ubuntu con Apache ocuparía unos 180 Mb (Nickoloff, 2016).

Desde el punto de vista del consumo de procesador y de memoria RAM, los contenedores Docker hacen un uso mucho más eficiente del sistema anfitrión, pues comparten con él, el núcleo del sistema operativo y parte de sus bibliotecas, con lo que únicamente usarán la memoria y la capacidad de cómputo que estrictamente necesiten (Nickoloff, 2016).

Docker provee un modo de ejecutar casi cualquier aplicación seguramente aislada en un contenedor. El aislamiento y la seguridad permiten ejecutar múltiples contenedores en el *host*, por lo que la naturaleza ligera de los contenedores se ejecuta sin la penalización extra de un *hypervisor*, lo que significa una mejor utilización del hardware disponible. El entorno que se desea desarrollar debe contar con un ambiente ligero, capaz de realizar técnicas de minerías de datos con el hardware que se encuentra disponible, necesitando a su vez que se procesen grandes volúmenes de datos con la rapidez y eficiencia necesaria para su realización. Para ello, la herramienta Docker constituye una excelente propuesta para desarrollar un entorno capaz de minar grandes volúmenes de datos, es por ello que se aplica su utilización en la propuesta de solución.

1.6 Motores de procesamiento de datos

1.6.1 MapReduce

El enfoque de MapReduce para el aprendizaje automático está basado en realizar el aprendizaje por lotes, en el que el conjunto de datos de capacitación se lee en su totalidad para crear un modelo de aprendizaje. El mayor inconveniente de este modelo de lotes es la falta de eficiencia en términos de velocidad y recursos computacionales (Sara Landset, 2015).

El mecanismo de tolerancia a fallas empleado por MapReduce se logra a través de la replicación de datos, el cual puede afectar la escalabilidad al aumentar aún más el tamaño de los datos. Se ha detectado que la necesidad de replicación de datos es responsable del 90% del tiempo de ejecución de las tareas de aprendizaje automático en MapReduce, y es quizás, el mayor impedimento para el procesamiento rápido de datos. Otra deficiencia de MapReduce, es que no permite fácilmente el procesamiento iterativo, lo que lo hace inadecuado para muchos proyectos de aprendizaje automático (Sara Landset, 2015).

1.6.2 Storm

Storm se utiliza para procesar datos en tiempo real y se concibió inicialmente para superar las deficiencias de otros procesadores en la recopilación y análisis de flujos de medios sociales. La arquitectura de Storm consiste en *spouts* y *bolts*, donde los *spouts* constituyen el flujo de entrada (por ejemplo, Twitter streaming API), mientras que los *bolts* contienen la mayor parte de la lógica de cálculo, procesando datos en forma de tuplas desde el *spouts* hacia otros *bolts*. Las redes de surtidores y los *bolts*, que se representan como gráficos dirigidos, se conocen como topologías (Sara Landset, 2015).

La tolerancia a fallas se logra por medio de la topología: donde los *spouts* mantendrán los mensajes en sus colas de salida hasta que los *bolts* los reconozcan. Los mensajes continuarán siendo enviados hasta que sean reconocidos, momento en el cual serán eliminados de la lista. Un nodo maestro, conocido como Nimbus rastrea los nodos trabajadores. Si un nodo trabajador deja de funcionar, entonces Nimbus reasignará a los trabajadores a otro nodo. Nimbus también se encarga de asignar tareas a los trabajadores, similar a Jobtracker en MapReduce. La mayor diferencia es si el rastreador de trabajos deja de funcionar, se pierden todos los trabajos en ejecución, pero si Nimbus deja de funcionar, se reinicia automáticamente (Sara Landset, 2015).

1.6.3 H2O

H2O es un marco de código abierto que proporciona un motor de procesamiento paralelo, análisis, matemática y bibliotecas de aprendizaje automático, junto con herramientas de preprocesamiento y evaluación de datos, además, de ofrecer una interfaz de usuario basada en la web. El motor de H2O procesa los datos completamente en la memoria, utilizando múltiples métodos de ejecución, dependiendo de lo que sea mejor para el algoritmo utilizado. El enfoque general utilizado es Distributed Fork / Join, una técnica de dividir y vencer, que es confiable y adecuada para tareas masivamente paralelas. Este es un método que divide un trabajo en trabajos más pequeños que se ejecutan en paralelo, lo que da como resultado un equilibrio dinámico de carga de granularidad para trabajos de MapReduce, así como también gráficos y flujos (Sara Landset, 2015).

1.6.4 Flink

Flink ofrece capacidad para el procesamiento por lotes y por flujo. Tiene su propio tiempo de ejecución, en lugar de estar construido sobre MapReduce. Como tal, puede integrarse con HDFS y YARN, o ejecutarse de forma completamente independiente del ecosistema de Hadoop. El modelo de procesamiento de Flink aplica transformaciones a colecciones de datos paralelas. Al igual que Spark, Flink también ofrece lotes iterativos y opciones de transmisión, aunque su API de transmisión se basa en eventos individuales, en lugar del enfoque de micro-lotes que utiliza Spark. Este es el mismo modelo que Storm usa para el verdadero procesamiento en tiempo real. Spark se presenta muy superior en las áreas de tolerancia a fallas y manejo de algoritmos iterativos, mientras que las ventajas de Flink radican en la presencia de mecanismos de optimización y una mejor integración con otros proyectos (Sara Landset, 2015).

1.6.5 Spark

Apache Spark es un sistema de computación en *cluster* de propósito general para procesar cargas de trabajo de minería de datos. Lo que diferencia a Spark de sus predecesores, como MapReduce, es su velocidad, facilidad de uso y análisis sofisticados. En términos de velocidad se puede asegurar que Spark puede lograr una latencia por debajo del segundo en cargas de trabajo, para ello hace uso de la memoria para el almacenamiento. En el caso de MapReduce, la memoria se usa principalmente para el cálculo real, mientras que Spark usa memoria tanto para calcular como para almacenar objetos. Spark proporciona un amplio conjunto de bibliotecas de alto nivel para diferentes tareas de cómputo de minería de datos, tales como aprendizaje automático, procesamiento SQL, procesamiento de gráficos y transmisión en tiempo real. Estas bibliotecas hacen que el desarrollo

sea más rápido y se puedan combinar de forma arbitraria. Spark es un proyecto de comunidad de fuente abierta, y son muy utilizadas las distribuciones de Apache de fuente abierta para las implementaciones (Rodríguez P. M., 2015) (Yadav, 2015).

Una aplicación Spark implica cinco entidades claves: un programa de controlador, un administrador de *cluster*, trabajadores, ejecutores y tareas. Un trabajador proporciona recursos de CPU, memoria y almacenamiento a una aplicación Spark. Los trabajadores ejecutan una aplicación Spark como procesos distribuidos en un grupo de nodos. Por su parte, el administrador de *cluster*, como su nombre lo indica, administra los recursos informáticos en un grupo de nodos trabajadores. Proporciona además, una programación de bajo nivel de recursos de *cluster* en todas las aplicaciones y permite que varias aplicaciones compartan recursos del *cluster* y se ejecuten en los mismos nodos trabajadores. Un programa controlador (*Driver Program*) es una aplicación que utiliza Spark como una biblioteca, la cual proporciona el código de procesamiento de datos que Spark ejecuta en los nodos trabajadores, donde un programa controlador puede iniciar uno o más trabajos en un *cluster* Spark. El ejecutor, es un proceso JVM (máquina virtual de Java o *Java Virtual Machine*) que Spark crea en cada nodo trabajador para una aplicación, el cual ejecuta el código de la aplicación al mismo tiempo que múltiples hilos y que también puede almacenar datos en la memoria o el disco. Un ejecutor tiene la misma duración de vida que la aplicación para la que se creó, cuando una aplicación Spark finaliza, todos los ejecutores creados para él también terminan. Las tareas (*Task*), son la unidad más pequeña que Spark envía a un ejecutor, donde cada tarea realiza algunos cálculos para devolver un resultado a un programa de controlador o particiona su salida para mezclar. Spark crea una tarea por partición de datos, y un ejecutor establece una o más tareas al mismo tiempo. La cantidad de paralelismo está determinado por el número de particiones, donde más particiones significan más tareas de procesamiento de datos en paralelo. Lo anteriormente planteado se ilustra en la figura 16 (Guller, 2015).

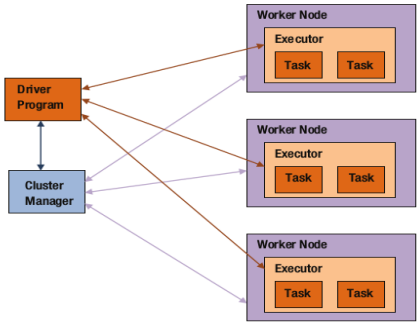


Figura 16: Arquitectura de Spark de alto nivel (Guller, 2015).

Las abstracciones principales utilizadas por Spark se denominan Datasets Distribuidos Resistentes (por sus siglas en inglés RDD), que almacenan datos en la memoria y proporcionan tolerancia a fallas sin replicación, los que se pueden entender como memoria compartida distribuida de solo lectura (Sara Landset, 2015).

Los RDD representan una colección de elementos de datos particionados que se pueden operar en paralelo. Es el mecanismo principal de abstracción de datos en Spark, y se definen como una clase abstracta en la biblioteca Spark. Los RDD están diseñados para ser tolerantes a fallas, ya que

representan los datos distribuidos en un *cluster* de nodos, previendo que un nodo pueda fallar. La probabilidad de que un nodo falle es proporcional al número de nodos en un *cluster*, cuanto más grande es un *cluster*, mayor es la probabilidad de que algún nodo falle. Los RDD manejan automáticamente las fallas de nodo, cuando un nodo falla y las particiones almacenadas en ese nodo se vuelven inaccesibles, Spark reconstruye las particiones RDD perdidas en otro nodo. Spark almacena información de linaje para cada RDD, usando esta información puede recuperar las partes de un RDD o incluso un RDD completo en caso de fallas de nodo (Guller, 2015).

1.7 Análisis de los motores de procesamiento de datos

Para una visión completa de cómo las herramientas y los motores con que se relacionan, la figura 17 se ilustra las relaciones entre los motores de procesamiento, los marcos de aprendizaje automático y los algoritmos que implementan (Sara Landset, 2015).

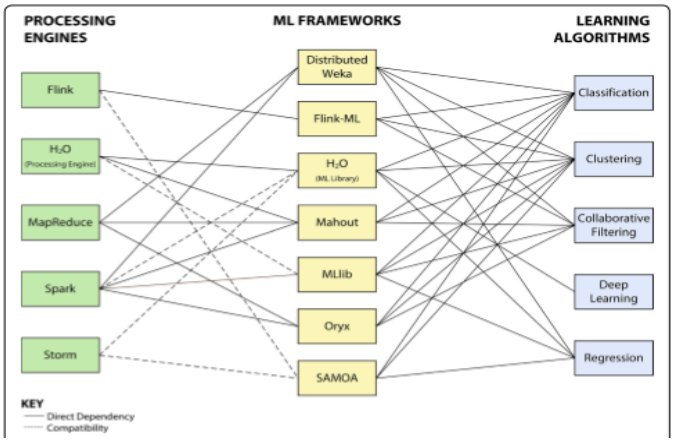


Figura 17: Marcos de aprendizaje de máquinas y sus motores de procesamiento asociados y algoritmos de aprendizaje (Sara Landset, 2015).

Una comparación de los marcos de aprendizaje de máquina es mostrada en la figura 18 (Sara Landset, 2015).

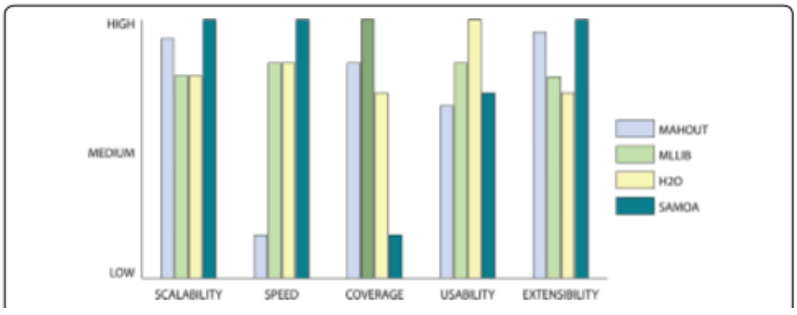


Figura 18: Comparación de marcos de aprendizaje de máquina (Sara Landset, 2015).

MLlib y H2O son muy buenas opciones para las necesidades generales; cada uno es rápido, escalable para diferentes tamaños de conjuntos de datos y tienen una selección bastante diversa de algoritmos. MLlib ofrece una mejor selección para la mayoría de las áreas de aprendizaje automático, pero H2O es la única herramienta que tiene soluciones para el aprendizaje profundo. En términos de usabilidad, ambos tienen API para programar en múltiples idiomas. MLlib cubre la

misma gama de categorías de aprendizaje que Mahout, y también agrega modelos de regresión que Mahout no tiene, además de que también posee algoritmos para el modelado de temas de minería de patrones. En general la confianza de MLlib en los enfoques iterativos de lotes y transmisión por secuencias de Spark, así como su uso de memoria computacional, permite que los trabajos se ejecuten significativamente más rápido que los que utiliza Mahout. SAMOA y Mahout (a través del nuevo Mahout-Samsara) se enfocan en ofrecer plataformas a través de las cuales el usuario puede crear sus propias implementaciones de algoritmos de aprendizaje. SAMOA permite al usuario crear implementaciones de algoritmos de transmisión, mientras que Mahout-Samsara se puede usar para implementaciones por lotes. Mahout y MLlib son las bibliotecas de minería de datos más completas en términos de cobertura de algoritmos y ambas funcionan con Spark y H2O, en la que MLlib tiene una selección general más amplia de algoritmos (Sara Landset, 2015).

El aprendizaje eficiente a partir de diferentes datos provenientes de diversas fuentes requiere arquitecturas complejas que utilizan combinaciones de herramientas y técnicas para la recopilación, el almacenamiento, el procesamiento y el análisis. En términos generales se puede arribar a la conclusión que para el desarrollo de la presente investigación se utilizarán las herramientas Spark haciendo uso del *framework* MLlib, para el procesamiento de datos heterogéneos utilizando algoritmos de minería de datos. Es seleccionada esta herramienta producto a la gama de características y ventajas analizadas que ofrecen con su aplicación, propiciando la eficiencia que se desea alcanzar en el entorno distribuido que se pretende desarrollar.

1.8 Gestión de datos

1.8.1 Hadoop

Cada día, se crean 2,5 quintillones de datos, donde el principal problema al que se enfrentan las instituciones es cómo convertir datos no estructurados a datos estructurados. El concepto de datos no estructurados es referido a la información que está dispersa y no está organizada de manera adecuada, por el contrario los datos estructurados juega un papel vital para extraer alguna información importante del conjunto de datos (Mr. Piyush Bhardwaj A. G., 2016). Para convertir los datos no estructurados en estructurados existen dentro del área de la minería de datos herramientas que apoyan este proceso, dentro de las más conocidas se encuentran Horton Works, Cloudera, Pivotal HD, Map Reduce e IBM Infosphere BigInsights, todas distribuciones de Hadoop (Allae Erraissi, 2017)

El objetivo principal de Hadoop es el almacenamiento de sets de datos de gran volumen en un *cluster* de nodos desplegados en hardware convencional. El núcleo de Hadoop está basado en una parte de almacenamiento definido como HDFS (*Hadoop Distributed File System*), y otra parte para el procesamiento definido como Map Reduce. Hadoop tiene entre sus principales características que divide ficheros en grandes bloques y los distribuye entre los nodos del *cluster*. En el caso del procesamiento a través de Map Reduce, el mismo transfiere el código empaquetado a los nodos para realizar el procesamiento en paralelo, teniendo en cuenta los datos que cada nodo debe procesar. Esta acción permite que los nodos manipulen los datos con los que dispone, asegurando

que el procesado se ejecute de forma más rápida y eficiente que una estructura que utilice supercomputación, asegurando con esta estructura que se dependa de un sistema de ficheros paralelizado donde la computación y los datos están aislados y conectados por redes de alta velocidad (Rodríguez P. M., 2015) (Mr. Piyush Bhardwaj A. G., 2016) (Chuck, 2011) (White, 2009) (Deepika P, 2015).

Hadoop está basado en el lenguaje de programación Java y los módulos que definen esta herramienta son (Prachi Pandey, 2016):

- ✓ Hadoop Common
- ✓ Hadoop Distributed File System (HDFS)
- ✓ Hadoop YARN
- ✓ Hadoop MapReduce

El sistema de ficheros HDFS se describe como escalable, distribuido y portable que implementa la estructura maestro/esclavo. Un *cluster* en Hadoop está compuesto por un NameNode que describe un servidor maestro que gestiona el espacio de nombres del sistema de archivos y regula el acceso a los mismos, y varios DataNodes, generalmente uno por nodo en el *cluster* que gestiona el almacenamiento a los nodos en los que se ejecutan (Lam, 2011) (S. Honnutagi, 2014) (V. Sajwan, 2015).

1.8.1.1 Sistemas de fichero

HDFS crea múltiples réplicas de cada bloque de datos que son distribuidas en *clusters* para permitir el acceso rápido y de forma fiable de la información. En este aspecto se implementan los componentes (S.Honnutagi, 2014) (R.Vijayakumari, 2014) (V. Sajwan, 2015) (B.Thillaieswari M.S., 2017):

- ✓ NameNode: Se describe como el componente principal del sistema, ocupando la responsabilidad de nodo *master*, encargado de mantener los nombres del sistema de archivos y administrar los bloques presentes en los DataNodes.
- ✓ DataNodes: Se describe como los nodos esclavos o trabajadores que son desplegados en los nodos del *cluster*, los cuales son los encargados de atender peticiones de lectura y escritura desde los clientes del sistema de archivos.

La figura 19 que se presenta a continuación muestra la arquitectura empleada por el HDFS (S. Honnutagi, 2014) (R.Vijayakumari, 2014):

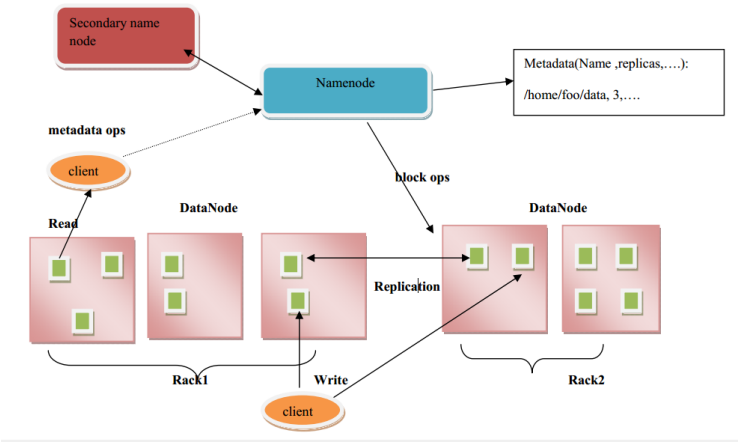


Figura 19: Arquitectura del HDFS (S. Honnutagi, 2014) (R.Vijayakumari, 2014).

1.8.2 IBM Infosphere

La plataforma InfoSphere se utiliza para gestionar los datos que son muy importantes en cualquier negocio, incluye varios módulos funcionales como integración de datos, almacenamiento de datos, gestión de datos maestros, minería de datos y gobernanza de la información. La plataforma proporciona una base de clase empresarial para diferentes proyectos de minería de datos, proporcionando rendimiento, escalabilidad, confiabilidad y precisión necesarios para la entrega de información. IBM InfoSphere DataStage es una herramienta de ETL[1] y constituye una parte del conjunto de soluciones IBM InfoSphere e IBM Information Platforms. Es muy útil para construir varias soluciones de integración de datos y tiene varias versiones que incluyen Server Edition, Enterprise Edition y MVS Edition (Mr. Piyush Bhardwaj A. G., 2016). A continuación se muestra la estructura de dicha plataforma (Allae Erraissi, 2017).

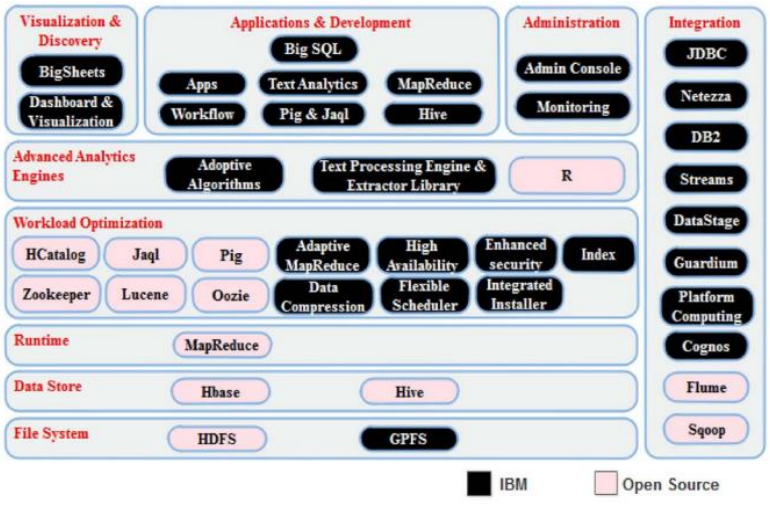


Figura 20: Funcionamiento de la plataforma IBM InfoSphere BigInsights Entreprise Edition (Allae Erraissi, 2017).

[1] ETL: Extract, Transform and Load (extraer, transformar y cargar) enmarcan dentro de las actividades claves en el contexto de las bases de datos, permiten hacer el traslado de datos de una fuente a otra (Néstor D. Duque Méndez, 2016).

.

1.8.3 Horton works distribution

En 2011, el equipo de Yahoo a cargo del proyecto Hadoop formó la HortonWorks. Todos los componentes de esta distribución son de código abierto y con licencia de Apache para facilitar la adopción de la plataforma Apache Hadoop. HortonWorks es un gran colaborador de Hadoop, y su modelo económico no tiene como propósito vender una licencia, sino vender exclusivamente soporte y formación, la figura que se muestra a continuación hace referencia a su estructura (Allae Erraissi, 2017).

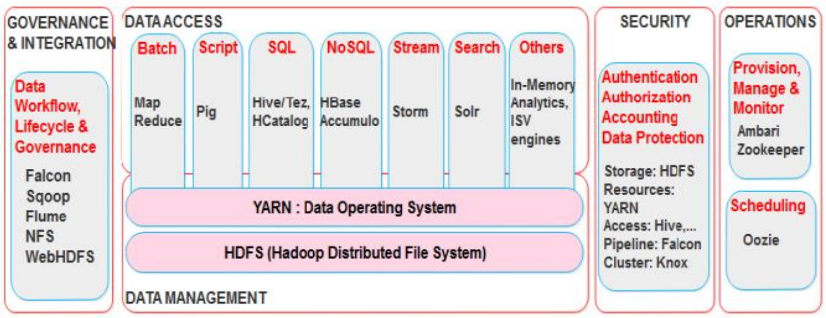


Figura 21: Funcionamiento de HortonWorks Hadoop Platform (HDP) (Allae Erraissi, 2017).

1.8.4 Pivotal HD distribution

Pivotal Software, Inc. (Pivotal) es una compañía de software y servicios. Incluye divisiones (Pivotal Labs) para servicios de consultoría, un grupo de desarrollo (Pivotal Cloud Foundry) y un grupo de desarrollo de productos para el mercado Big Data. Pivotal HD Enterprise es una distribución de paquetes Apache Hadoop compatible con la empresa y comercialmente orientada a las implementaciones tradicionales de Hadoop. A continuación es mostrada la estructura de funcionamiento de esta distribución (Allae Erraissi, 2017).

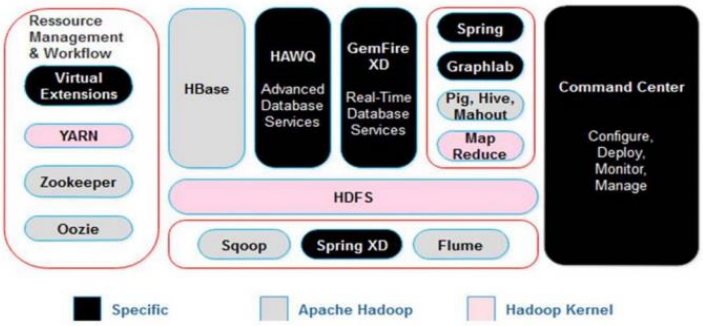


Figura 22: Funcionamiento de Pivotal HD Entreprise (Allae Erraissi, 2017).

1.8.5 Cloudera distribution

Cloudera fue fundado por expertos de Hadoop, Facebook, Google, Oracle y Yahoo. Esta distribución se basa en gran medida en los componentes de Apache Hadoop y se complementa con componentes para la administración de *clusters*. El objetivo del modelo de negocios de Cloudera no está basado en solo vender licencias, sino también en vender soporte y capacitación. A continuación es mostrado su funcionamiento (Allae Erraissi, 2017).

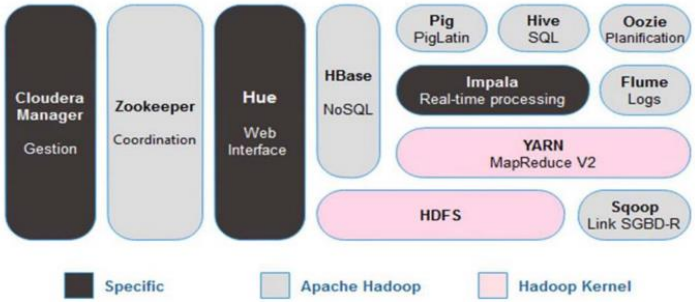


Figura 23: Funcionamiento de Cloudera Distribution for Hadoop Platform (CDH) (Allae Erraissi, 2017).

1.9 Comparación entre las distribuciones de Hadoop

Estableciendo una comparación entre las principales distribuciones de Hadoop, se puede arribar a la conclusión de que dichas distribuciones presentan diversas características que permiten afirmar que no existe un ganador absoluto, ya que cada proveedor se enfoca en las características principales dedicadas a los sistemas de minería de datos, tales como integración, seguridad y escalabilidad (Allae Erraissi, 2017).

Hadoop HDFS se ajusta a la problemática planteada de garantizar una adecuada eficiencia en el entorno distribuido que se desea desarrollar, posee características de gran almacenamiento de conjuntos de datos para propiciar el procesamiento de información, basado en el lenguaje de programación Java. Es una herramienta que utiliza *commodity hardware*, lo que se traduce en que no necesita de hardware de grandes prestaciones con gran capacidad de memoria RAM para el procesamiento de datos, es posible realizar todas estas operaciones con hardware de bajo costo. Además, la utilización de esta herramienta contribuye a la creación de un entorno distribuido robusto que cumple con características como la tolerancia a fallas, la escalabilidad y el alto rendimiento.

Conclusiones del capítulo:

- ✓ El análisis de los conceptos Minería de Datos, Minería de Datos Educativos y Minería de Datos Distribuida permitió identificar los elementos necesarios para el desarrollo de un entorno distribuido, para ser aplicado a modelos de estimación del conocimiento latente en datos educacionales masivos.
- ✓ Las herramientas Hadoop HDFS para el proceso de gestión de datos, Spark mediante el *framework* MLlib, para el procesamiento de datos heterogéneos y Docker para la virtualización del procesamiento de los datos, propician el desarrollo de un entorno distribuido que garantice una adecuada eficiencia.
- ✓ El análisis de los entornos distribuidos permitió identificar las características de transparencia, escalabilidad y tolerancia a fallos como esenciales en el desarrollo del entorno distribuido para el minado de datos en ambientes educacionales masivos.

2 CAPÍTULO 2 : DESARROLLO DEL ENTORNO DISTRIBUIDO PARA EL MINADO DE DATOS EN AMBIENTES EDUCACIONALES MASIVOS.

En el capítulo que se expone a continuación se define la arquitectura por el que será desarrollado el entorno distribuido. Se realiza la descripción de la propuesta de solución y se detalla el flujo de los procesos involucrados garantizando a través de su utilización el cumplimiento de la hipótesis planteada al inicio de la presente investigación.

2.1 Modelo para la estimación del conocimiento latente en datos educacionales masivos

Constantemente usuarios tales como: instructores, estudiantes y administradores de cursos interactúan con sistemas educativos informatizados como son las aulas inteligentes, sistemas *e-learning*, LMS (*Learning Management System* o Sistema de Gestión de Aprendizaje) entre otros. Los mismos generan un gran cúmulo de información que es registrada en bases de datos educacionales. A partir de la generación de estas bases de datos son aplicadas las técnicas de minería de datos, las cuales funcionan a través de algoritmos de inteligencia artificial, haciendo posible la identificación de patrones de comportamiento para la estimación del conocimiento latente. La figura 24 muestra un esquema general de lo antes descrito (S. Aghabozorgi, 2014).

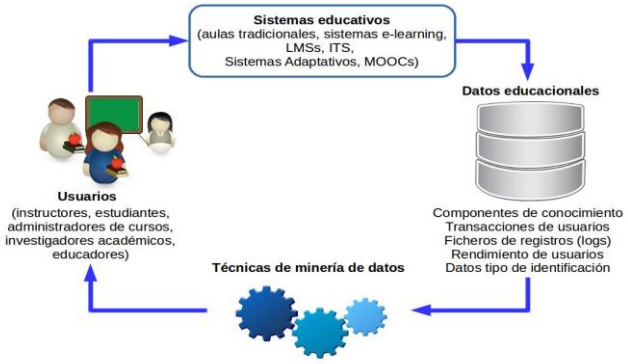


Figura 24: Proceso para la aplicación de la minería de datos en el contexto educativo (S. Aghabozorgi, 2014).

Una de las principales tareas de la minería de datos es la predicción, cuyo objetivo es desarrollar modelos que permitan inferir un aspecto simple de los datos. En este sentido, la minería de datos educacional, los clasificadores y las regresiones son los modelos de predicción más comunes. El área de la estimación de conocimiento latente (ECL) es de particular importancia dentro de la MDE, donde el trabajo en esta área se debe al surgimiento del modelado de usuarios, la inteligencia artificial utilizada en el sector educacional, y las métricas psicométricas/educacionales. En la estimación del conocimiento latente, el dominio de un estudiante sobre determinadas habilidades y conceptos es evaluado por sus patrones de corrección de esas habilidades. El término “latente” es referido a la idea de que el conocimiento no puede ser medido directamente, sino inferido a través del rendimiento del estudiante (Johann Ari Larusson, 2014). Incrementar el conocimiento de los estudiantes es la principal meta de la educación, es por ello que si el conocimiento puede ser

inferido, se puede determinar de qué forma aumenta la capacidad de aprendizaje del estudiante, y por tanto ejecutar decisiones pedagógicas automáticas (Corbett, 2014).

En este sentido, se ha desarrollado en la Facultad 4 de la Universidad de las Ciencias Informáticas un modelo capaz de estimar el conocimiento latente del estudiante a partir de datos educacionales masivos. Dicho modelo será aplicado teniendo en cuenta el contexto de la educación cubana y los proyectos que se realizan con el fin de la informatización de nuestro país. La estructura que sigue dicho modelo, se muestra en la figura 25 que se muestra a continuación.

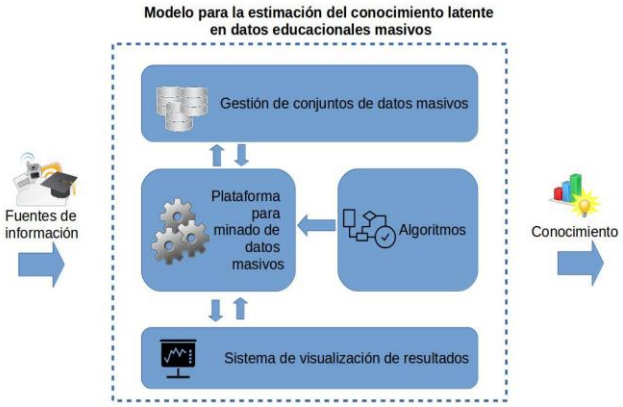


Figura 25: Modelo para la estimación de conocimiento latente en datos educacionales masivos, Fuente: Proyecto GITAE Línea MDEM.

El modelo está compuesto por cuatro fases, dando inicio a través de las diversas fuentes de información educacionales. Los datos educacionales almacenados son agrupados para formar un gran conjunto, el cual es tratado en el primer proceso que conforma el modelo de la figura 25, que describe el proceso de gestión de conjunto de datos masivos. Posteriormente los datos obtenidos, se trasladan hacia el segundo proceso, denominado plataforma para el minado de datos masivos, la cual interactúa directamente con los algoritmos de minería de datos a utilizarse, en la que los resultados del procesamiento son trasladados al sistema de visualización. De esta manera a través de técnicas predictivas es obtenida la estimación del conocimiento.

La plataforma para el minado de datos masivos, ocupa gran relevancia dentro del modelo. Esta plataforma, es la encargada de recibir los datos del proceso anterior y realizar el agrupamiento de dichos datos. Posteriormente, se aplican dentro de la plataforma los algoritmos, asegurando a su vez que los mismos se ejecuten de forma eficiente teniendo en cuenta el hardware disponible. Finalmente, es obtenida la estimación del conocimiento, la cual es visualizada en el siguiente proceso del modelo descrito anteriormente. Debido a la importancia que ocupa la plataforma para el minado de datos masivos dentro del modelo desarrollado y la importancia que se le confiere a la velocidad de procesamiento de los datos dentro de la misma, se genera la necesidad de la presente investigación, de conformar un entorno distribuido para contribuir al uso de técnicas de minería de datos distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos.

2.2 Arquitectura distribuida para el entorno desarrollado

Como parte de la solución propuesta, se plantearon con anterioridad conceptos tales como minería de datos, que está basado en la aplicación de algoritmos y técnicas en fuentes de datos heterogéneas, para a partir de las mismas extraer conocimiento. Además se describió el concepto de minería de datos educacional, que se define como un área que emerge dentro de la minería de datos, la cual está centrada en el desarrollo de métodos para hacer descubrimientos dentro de tipos de datos que provienen de entornos educativos, y el uso de esos métodos para comprender mejor al estudiante y la forma en el que el mismo aprende.

Las técnicas de minería de datos emplean gran procesamiento y necesitan una completa robustez tecnológica que brinde soporte a los sistemas encargados de extraer conocimiento. Para ello, la minería de datos distribuida se convierte en un punto clave de la presente investigación, para a partir de la misma crear un entorno que propicie la aplicación de diversos algoritmos que utilicen minería de datos educacional. La aplicación de la MDD ofrece diversas ventajas, como es el caso de utilizar una estrategia descentralizada haciendo posible que el sistema de trabajo sea completamente escalable, logrando distribuir la carga de trabajo entre diversas estaciones de forma heterogénea. Como valor agregado, esta estrategia posibilita la reducción de costos que presupone la utilización de grandes servidores dedicados a la extracción y procesamiento de los datos.

A partir de la utilización de la minería de datos distribuida en el desarrollo del entorno, se estableció una arquitectura que es capaz de sostener las bases por la cual se aplica esta técnica en la propuesta de solución. Dicha arquitectura sienta sus bases en los elementos básicos que compone todo enfoque basado en la MDD y está compuesta por cuatro niveles.

Para el primer nivel se tuvo en cuenta factores tales como la distribución de diferentes fuentes de datos educacionales, para en un segundo nivel realizar su unificación a través de una vista integrada de datos. A partir de la vista integrada de datos confeccionada se ejecuta el tercer nivel, en el cual se realizará la implementación de los algoritmos que funcionan dentro del entorno. Se aplicará algoritmos tales como el Rastreo Bayesiano del Conocimiento (más conocido por *Bayesian Knowledge Tracing*, o BKT por sus siglas en inglés), la Teoría de Respuesta al Ítem (más comúnmente conocido *por Item Response Theory*, IRT) y el Algoritmo de Factor de Rendimiento, PFA por sus siglas en inglés (*Performance Factor Algorithm*). Todos estos algoritmos tendrán que funcionar de forma distribuida en nodos dispersos, para propiciar un alto grado de procesamiento de datos de forma eficiente haciendo uso del hardware disponible.

Los resultados obtenidos por los algoritmos serán expuestos en un modelo global en el cuarto nivel, haciendo uso de las llamadas curvas de aprendizaje, las cuales infieren el conocimiento estimado que ostenta un estudiante sobre un tema en particular (Perlich, 2011). Es decir, el estudiante tendrá la posibilidad de visualizar por tres algoritmos diferentes el conocimiento estimado que posee. A continuación en la figura se muestra la arquitectura elaborada para el desarrollo del entorno.

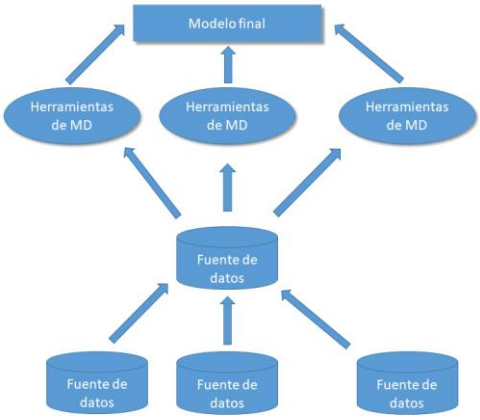


Figura 26: Arquitectura generada para el entorno distribuido para el minado de datos en ambientes educacionales masivos. Fuente: Elaboración propia.

2.3 Flujo de procesos del entorno distribuido para el minado de datos educacionales masivos

Para la confección del entorno fue necesario tener en cuenta los procesos que intervienen en la creación del mismo. Se identificaron las fuentes de datos que se utilizan como entrada para el proceso de minado que se realiza dentro del entorno y los datos de salida que necesita generar el mismo.

Es importante tener en cuenta los subprocesos asociados en cada proceso para garantizar el cumplimiento a la hipótesis planteada en la investigación que se desarrolla. Para ello se cuenta con un total de tres procesos, generándose a partir de la entrada de fuentes de datos y concluyendo con la obtención del conocimiento estimado. La figura 27 que se muestra a continuación, representa el flujo de procesos del entorno distribuido.

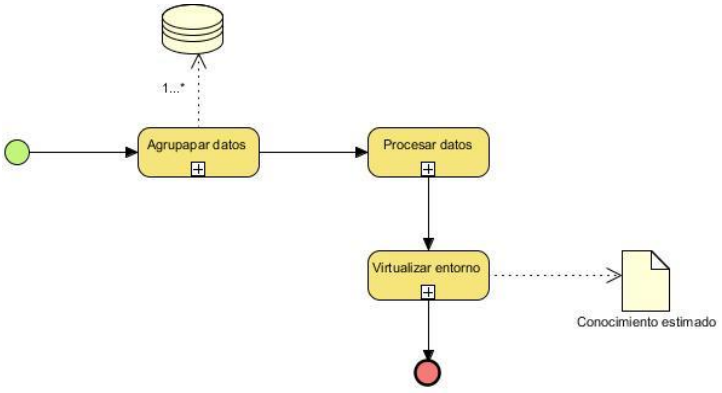


Figura 27: Diagrama de flujo: Entorno distribuido para el minado de datos en ambientes educacionales masivos. Fuente: Elaboración propia.

2.3.1 Proceso de agrupamiento de datos a través de Hadoop HDFS

Para el proceso de agrupamiento de datos se hará uso de la herramienta Hadoop, basado en el sistema Hadoop Distributed File System o comúnmente llamado Hadoop HDFS, debido a las características que se plantearon con anterioridad, que contribuyen al proceso de agrupamiento de

datos masivos. Hadoop HDFS destaca entre sus principales características su diseño flexible en cuanto a costos, por lo que la aplicación del *commodity hardware*, como también es conocido, beneficia a las organizaciones de implementar gran procesamiento de datos utilizando pocos recursos. Otra característica importante que posee esta herramienta, es que proporciona una alta tolerancia a fallas, alto rendimiento y escalabilidad, propiedades que debe cumplir todo sistema distribuido para su explotación, además de realizar todas estas operaciones con hardware de bajo costo.

El proceso de agrupamiento de datos está compuesto por varios subprocesos, como es mostrado en el diagrama de flujo de la figura 28.

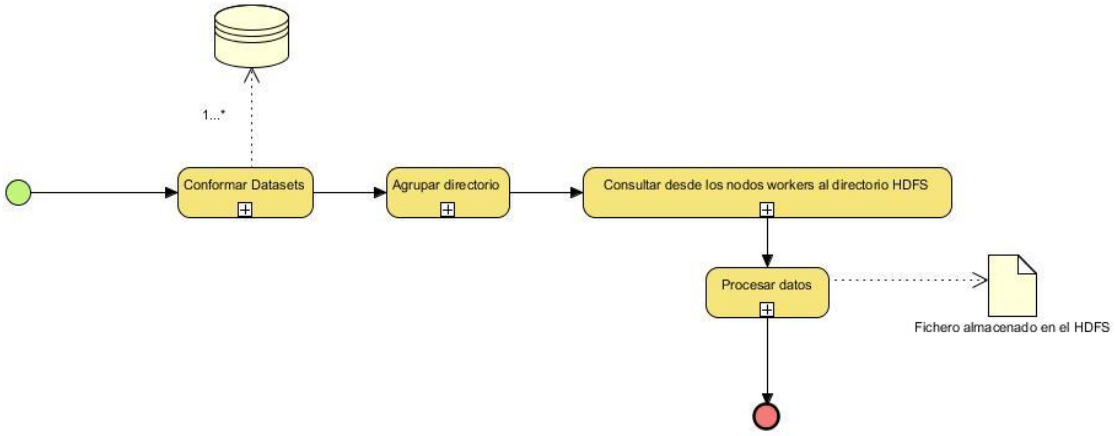


Figura 28: Diagrama de flujo: Agrupar datos. Fuente: Elaboración propia.

El inicio del mismo, se realiza a partir de la entrada de bases de datos distribuidas provenientes del proceso anterior que describe el modelo para la estimación del conocimiento latente en datos educacionales masivos, las cuales son accedidas a través de una dirección URL, realizándose el proceso de forma virtual para la conformación de los *datasets*. HDFS crea múltiples réplicas de cada bloque de datos y las distribuye en clústeres para permitir el acceso rápido y fiable. Este subproceso está dado en que a partir de bases de datos heterogéneas distribuidas llamadas Data Nodes, sean agrupadas en un solo nodo llamado *Name Node*. Hadoop HDFS posee una arquitectura maestro/esclavo, en el que un *cluster* posee un solo *Name Node* en el servidor maestro que gestiona el espacio de nombres del sistema de archivos y regula el acceso a los mismos.

Para realizar el flujo de trabajo del establecimiento de los *clusters* se sigue el siguiente orden:

- ✓ Selección e instalación de la versión de Hadoop a utilizarse.
- ✓ Definición de la configuración del *cluster*.
- ✓ Operar el *cluster* desplegado para el tratamiento con los nodos.

El primer punto del flujo de trabajo que se establece, define la selección de la versión de la herramienta Hadoop, en este caso será utilizada la versión 2.8.0. Para la definición de la configuración se establece el tamaño que tendrá dicho *cluster* y la topología que establece el mismo, en el cual se define un total de tres nodos esclavos y un *master*, lo que se traduce de un total de tres *Data Nodes* y un *Name Node*. Posteriormente se realizarán todas las operaciones necesarias

para el tratamiento con los nodos, añadiendo y eliminando los mismos en caso de ser necesario. La figura 29 que se muestra a continuación refleja la instalación y configuración de la herramienta analizada.

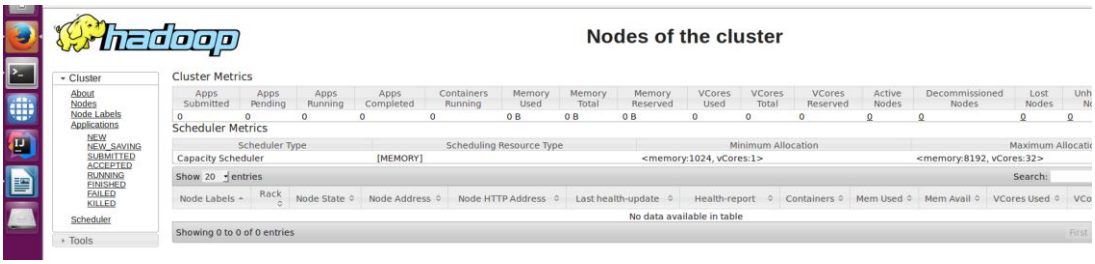


Figura 29: Entorno de trabajo con la herramienta Hadoop. Fuente: Elaboración propia.

Para la conformación de los *datasets* como se abordó anteriormente, se siguió una arquitectura maestro/esclavo a través de los *Data Nodes* y el *Name Node*. Las bases de datos distribuidas son accedidas desde el *Name Node* a través de la dirección URL de forma virtual proporcionada por los *Data Nodes* como es mostrado en la figura 30.

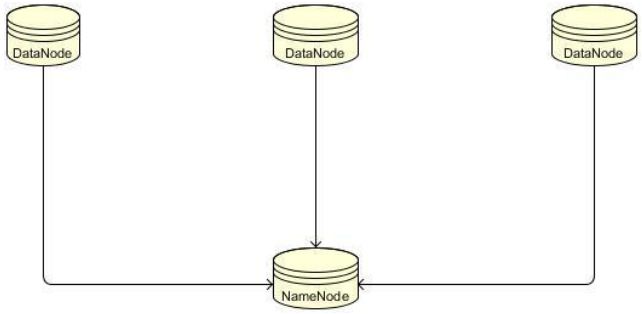


Figura 30: Conformar Datasets. Fuente: Elaboración propia.

El nodo *master* almacena el *dataset* de forma virtual conformado por las direcciones proporcionadas por los *Data Nodes*. La figura 31 que se presenta a continuación refleja esta arquitectura.

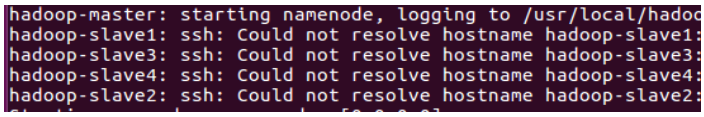


Figura 31: Estructura maestro/esclavo utilizando Hadoop. Fuente: Elaboración propia.

En el proceso de agrupar de directorio, se crea un directorio en el HDFS del nodo *master*, el cual aglutina todos los algoritmos que serán utilizados. Cuando se haga referencia a cualquiera de estos algoritmos se accederá al HDFS a través de la dirección URL donde se encuentra y el *dataset* correspondiente. La figura 32 que se presenta a continuación refleja el proceso descrito.

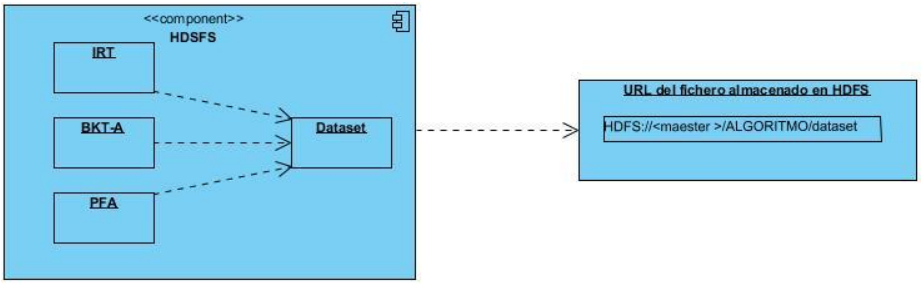


Figura 32: Agrupar directorio. Fuente: Elaboración propia.

El proceso de agrupar directorio trae consigo que a través de la URL que es generada, se realice el próximo proceso que conforma el flujo de agrupamiento de datos. Para realizar el proceso de consulta desde los nodos trabajadores al directorio HDFS, los nodos acceden a la dirección URL proporcionada por el *master* que hace referencia al algoritmo que se desea utilizar en conjunto con el *dataset* seleccionado. Cada nodo trabajador puede acceder a la URL proporcionada por el *master*, para a partir de la misma realizar el procesamiento distribuido de los datos. El resultado del procesamiento ejecutado por cada nodo trabajador, es almacenado en un fichero ubicado en el HDFS en el nodo *master*, el cual es accedido a través de la dirección URL.

De forma general se puede arribar a la conclusión que el proceso de agrupar datos dentro del entorno, aporta escalabilidad, transparencia y tolerancia a fallos, a través del uso de la herramienta Hadoop HDFS. La estructura descentralizada que se utiliza en la arquitectura que se establece, hace posible el cumplimiento de las características antes mencionada, además de facilitar la distribución de los recursos para garantizar el procesamiento distribuido de los datos de forma eficiente.

2.3.2 Procesamiento de datos a través de Spark

Para el procesamiento de datos se sigue la estructura antes mencionada maestro/esclavo, logrando mediante su utilización que sean ejecutados los procesos de minería de datos distribuida. Se establecen tres nodos esclavos y un nodo *master*, encargado este último de administrar y controlar todos los procesos ejecutados en los nodos trabajadores.

Teniendo en cuenta el fichero obtenido en el proceso anterior, en el cual se obtiene la dirección URL que contiene el algoritmo que se desea ejecutar con el *dataset* correspondiente, se procede al procesamiento de dichos datos siguiendo el flujo que se muestra en la figura 33.

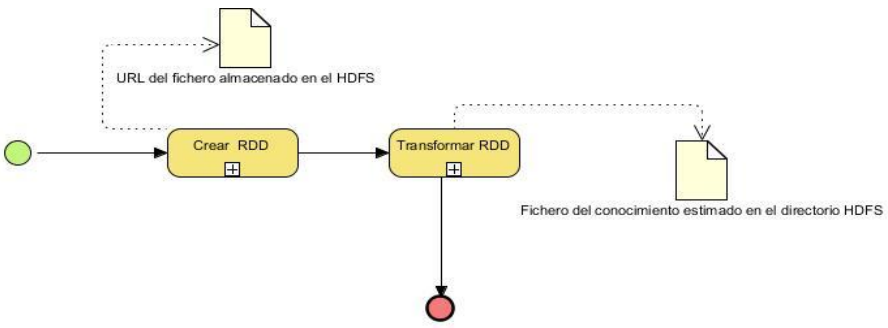


Figura 33: Diagrama de flujo: Procesar datos. Fuente: Elaboración propia.

El primer proceso del flujo describe la creación de los RDD, donde fue necesario apoyarse en el método *textFile* que define la herramienta Spark, el cual crea un RDD a través del archivo de texto que es leído del directorio almacenado en el HDFS. La línea de código que se muestra en la figura 34, crea un RDD desde el archivo ubicado en el directorio HDFS basado en el lenguaje Scala, utilizado por Spark para la implementación.

```
var rdd = sc.textFile("hdfs://<master:3710>/bkt/datasets/dataset1.txt");
```

Figura 34: Crear RDD utilizando el método textFile. Fuente: Elaboración propia.

Los datos representados en el RDD son divididos en particiones, las cuales son distribuidas en el *cluster* de nodos. Con la creación de los RDD se proporciona una abstracción para los datos que son almacenados en los nodos trabajadores. Es por ello que el HDFS almacena los datos en particiones o bloques que son distribuidos en estos nodos.

Luego de creados los RDD, se ejecuta el proceso de transformación de los mismos para crear un nuevo RDD a través de los cálculos realizados sobre el original. Las transformaciones realizadas en este sentido, están dadas a partir de los algoritmos identificados, como son el BKT, el IRT y el PFA. En este proceso es donde se establece la relación entre la plataforma de minado de datos y los algoritmos, los cuales constituyen partes fundamentales del modelo señalado en la figura 25.

Posteriormente de realizadas las transformaciones correspondientes, es obtenido un fichero que contiene el conocimiento estimado almacenado en el directorio HDFS. El fichero obtenido con el conocimiento estimado proveniente de los nodos trabajadores será almacenado en el nodo *master*, el cual puede ser consultado para su utilización a través de la dirección URL que es generada.

A continuación se muestra en la figura 35 el resultado del funcionamiento de la herramienta Spark con la arquitectura maestro/esclavo en el proceso de minería de datos.

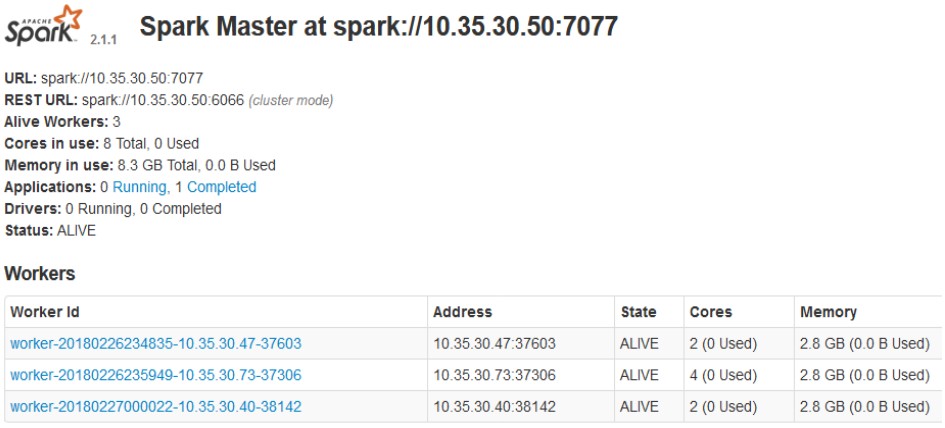


Figura 35: Proceso de minado de datos a través de Spark. Fuente: Elaboración propia.

A partir del flujo de trabajo establecido se garantiza el procesamiento en paralelo a través de los RDD. La creación de los RDD dentro del entorno permite que los datos puedan ser distribuidos en el *cluster*, para la realización de las transformaciones correspondientes. Es decir, que con la aplicación de una estrategia descentralizada para el análisis de datos, se puede garantizar que este proceso se realice de forma eficiente.

2.3.3 Virtualización a través de Docker

La virtualización ofrece características importantes en la creación del entorno que se desea desarrollar. La misma propicia un mejor aprovechamiento y ahorro del hardware disponible, lo que conlleva a una reducción de costes en cuanto a mantenimiento y administración. La herramienta de código abierto Docker, favorece la creación del entorno distribuido a través de las características que presenta. Mediante su utilización se obtiene un ambiente ligero para el proceso de minería de datos educacionales, la cual resulta de apoyo a la administración del entorno.

Para la utilización de Docker es necesario recordar que el mismo funciona a través de contenedores aislados, el cual se define como un proceso que se ejecuta en un *host*. Para su ejecución es necesario la aplicación del comando `docker run`, el cual debe especificar una imagen para derivar la ejecución de un contenedor. Una imagen se define como un paquete ejecutable que incluye todo lo necesario para ejecutar una aplicación, ya sea el código, un tiempo de ejecución, bibliotecas, variables de entorno y archivos de configuración.

La figura 41 que se presenta a continuación, reúne todo el proceso para la instalación y configuración de los nodos en el entorno, utilizando la herramienta Docker.

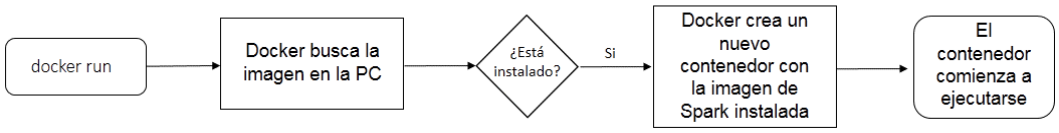


Figura 36: Ejecución de los contenedores. Fuente: Elaboración propia.

Un contenedor es una instancia de tiempo de una imagen, donde la imagen se convierte en memoria cuando se ejecuta. La imagen que se utilizará en el contenedor será la herramienta de software libre Spark en su versión 2.1.1, la cual se encargará del proceso de minado de datos. A continuación la figura 37 ilustra el contenedor desplegado con la imagen de Spark.

```
mdem@mdem-server ~> docker ps -a
CONTAINER ID        IMAGE
```

Figura 37: Contenedor de Docker con una imaen de Spark. Fuente: Elaboración propia.

El nodo *master*, realizará el proceso de clusterización a través de tres nodos trabajadores, que a su vez ejecutarán Docker Engine en un contenedor con la imagen de Spark. La figura 38 describe la arquitectura que tendrán los contenedores en cada nodo.

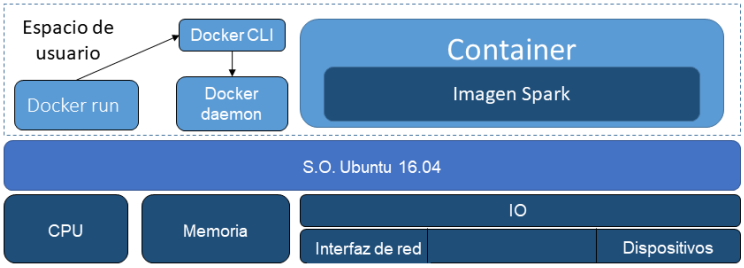


Figura 38: Estructura de los contenedores. Fuente: Elaboración propia.

Mediante la aplicación de Docker Swarm se puede asegurar el funcionamiento del *clustering*, utilizado para crear un grupo de nodos que proporcionen redundancia. Esta característica permite a través de su utilización que conmute el mismo por error si uno de los nodos trabajadores falla.

El diseño descentralizado que es aplicado a la propuesta de solución, tiene como resultado una diferenciación entre los roles de los nodos, en la que nodo *master* efectuará la orquestación de los nodos trabajadores. Los nodos trabajadores desarrollarán el proceso de minado de datos como se explicó anteriormente, implementando de forma distribuida y paralela los algoritmos BKT, IRT y PFA. A continuación se presenta en la figura 39 la arquitectura maestro/esclavo utilizada con una imagen de Spark.

```
c7451302bf1a        nexus.prod.uci.cu:1959/ spark-cluster:latest
0.0:8080->8080/tcp, 0.0.0.0:8088->8088/tcp, 0.0.0.0:50070->50070/tcp    spark-master
dbd2b85b7ad3        nexus.prod.uci.cu:1959/ spark-cluster:latest
                                                                        spark-slave2
0ac80104cfcc        nexus.prod.uci.cu:1959/ spark-cluster:latest
                                                                        spark-slave1
mdem@mdem-server -> docker port  spark-master
8088/tcp -> 0.0.0.0:8088
50070/tcp -> 0.0.0.0:50070
8080/tcp -> 0.0.0.0:8080
```

Figura 39: Arquitectura maestro/esclavo utilizada en Docker con una imagen de Spark. Fuente: Elaboración propia.

Haciendo uso de las características que trae implícita la utilización de Docker Swarm en el nodo *master*, se puede realizar la administración de los servicios en los nodos trabajadores con Docker Engine. La ejecución de Docker Swarm en el nodo *master*, permite orquestar todo lo referente a los nodos trabajadores sin tener que necesitar otro software adicional, posibilitando mediante su utilización un entorno controlado y administrado por sí mismo.

El nodo *master*, respaldará el número de tareas que se desea ejecutar en cada nodo trabajador, ya que el mismo se adapta automáticamente al agregar o eliminar tareas, lo que hace posible la escalabilidad del sistema. Cada nodo trabajador será supervisado constantemente por el nodo *master*, de forma tal que se conozca el estado real de cada nodo trabajador. Esta supervisión constante permite identificar si un nodo trabajador falla. En caso de ser identificada una falla en uno ellos, el *master* crea una nueva réplica para reemplazar el nodo que falló conmutando por error.

La figura 40 que se presenta a continuación, constituye la arquitectura con la cual se desarrolla la virtualización del entorno. La arquitectura está compuesta por un total de tres nodos trabajadores, en los que se ejecuta de forma paralela los algoritmos de minería de datos, y un nodo *master* capaz de orquestar y administrar todo lo relacionado con los nodos trabajadores.

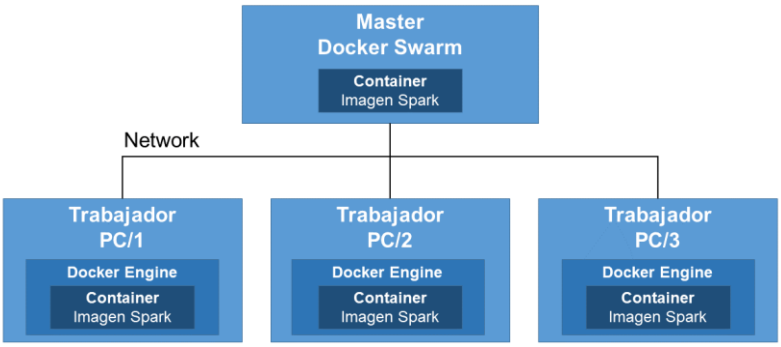


Figura 40: Arquitectura distribuida del entorno. Fuente: Elaboración propia.

La utilización de Docker dentro del entorno distribuido, propicia un ambiente ligero capaz de procesar grandes volúmenes de datos a gran velocidad a través de la herramienta Spark. Por otro lado, se puede afirmar que Docker Swarm a través del diseño distribuido que posee permite que el entorno sea escalable, transparente y tolerante a fallas.

2.4 Clustered data mining aplicado al entorno distribuido

A partir del concepto de *Clustered Data Mining* abordado con anterioridad, se aplicó el descubrimiento de conocimiento de diferentes fuentes de datos distribuidas en tres nodos, que funcionan como nodos autónomos interconectados a través de la herramienta Docker Engine. Dicha herramienta se ejecuta como un componente de aplicación paralela, en el que se combina los datos obtenidos en el nodo *master* a través de la herramienta Spark para obtener el conocimiento estimado que posee un estudiante. Los nodos trabajadores están acoplados de forma tal que ejecuten de forma simultánea los algoritmos de minado de datos educacionales. De esta forma, se asegura que el nodo *master* adquiera información de bases de datos heterogéneas y pueda ser administrada por el mismo.

El nodo *master* manejará la asignación de cada nodo, les dará seguimiento y controlará todo el proceso paralelo que se desarrolla en cada nodo trabajador. Docker Swarm aplicado al nodo *master*, tiene integrado un conjunto de funcionalidades que respaldan la administración de los nodos trabajadores donde son implementados servicios que permiten orquestar y administrar todo el sistema distribuido. Con la utilización de Docker Swarm el nodo *master* ejecuta el middleware (utilizado para proporcionar la visión de sistema único) necesario para la administración de los nodos trabajadores.

Los nodos trabajadores son clasificados como de clase *Beowulf* o como también es conocido *cluster* de productos, ya que el entorno propuesto está basado en componentes básicos de hardware de bajo costo y de software libre. Las herramientas Docker y Spark constituyen sistemas de software libre, mientras que el sistema operativo donde es implementado es Ubuntu con distribución Genome 16.04, sistema operativo libre que no incluye costos y propicia un entorno de trabajo robusto.

A continuación, en la figura 41 se muestra la arquitectura de trabajo descrita anteriormente.

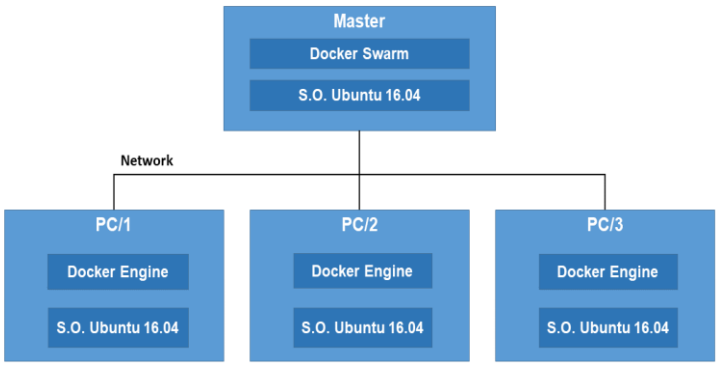


Figura 41: Arquitectura de tres cluster y un master del entorno distribuido. Fuente: Elaboración propia.

Es importante destacar que la tecnología de Docker funciona a través de la clusterización de los nodos trabajadores, los cuales se caracterizan por su homogeneidad en cuanto a hardware, sistema operativo y la red que utilizan. Esta aplicación trae consigo una mejor administración y centralización de la información en el nodo *master* a través de Spark, información que puede caracterizarse por ser sensible y que necesita de un correcto uso y seguridad.

2.5 Propiedades de los sistemas distribuidos aplicadas al entorno

Para que un sistema se considere distribuido, debe aglomerar un conjunto de computadoras que se encuentren interconectadas para ofrecer una visión de sistema único, mostrando los resultados de forma homogénea, ocultando su distribución al usuario y a las aplicaciones haciéndolo a su vez un sistema indistinguible. Dichos propósitos para su cumplimiento deben reunir ciertas características que deben ser retomadas para su aplicación en el entorno propuesto.

2.5.1 Transparencia

La transparencia del entorno, tiene entre sus características principales el hecho de que los procesos y los recursos que se encuentran distribuidos en tres nodos trabajadores sean totalmente ocultos. La utilización de Docker Engine en cada nodo trabajador, propicia la capacidad de que las aplicaciones y los usuarios sean tratados como un sistema único gestionados por un solo contenedor de Docker.

El *master* a través de Docker Swarm, monitorea continuamente el estado del grupo y reconcilia cualquier diferencia entre el estado real y el estado deseado expresado. A través de la configuración de un servicio para ejecutar tres réplicas de un contenedor y tres máquinas de trabajo que hospedan respectivamente estas réplicas, en caso de un fallo, el *master* creará una nueva réplica para reemplazar la que falló. Swarm Manager asigna una nueva réplica en los nodos trabajadores que se están ejecutando y están disponibles.

Para establecer una unión entre un nodo *master* y los nodos trabajadores se tiene que obtener el *token* de la PC que servirá como trabajador, para posteriormente ejecutar la siguiente línea de comando que se muestra en la figura 42, la cual establece el enlace entre nodo *master* y el nodo trabajador.

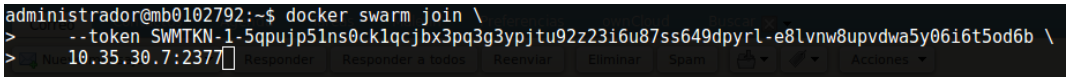


Figura 42: Unión entre el nodo master y nodo trabajador a través de un token. Fuente: Elaboración propia.

De esta manera, el nodo *master* tiene total acceso a los procesos que se realizan en los nodos que se desarrollen como trabajadores. Es decir, si falla o deja de ejecutarse alguno de los nodos trabajadores, el nodo *master* tendrá el control sobre el mismo y habrá realizado las réplicas de respaldo del nodo que falló sobre los nodos trabajadores que estén disponibles.

Con la utilización de Docker, la ubicación física de cada recurso que es procesada en cada nodo trabajador, es solamente accedida y gestionada por el nodo *master*. Esto significa, que cada nodo trabajador se ejecuta de forma paralela y no existe comunicación entre los mismos si no es accedido

a través del *master*. Esta cualidad, propicia que los recursos tengan la posibilidad de migrar al nodo *master* en caso de que exista una falla en un nodo trabajador, sin que se pierda la información que ha sido obtenida a través de los algoritmos.

Otra característica importante para lograr una alta disponibilidad en un sistema distribuido, es a través de la replicación. La arquitectura de la figura 45 muestra la ejecución de tres máquinas que funcionan de forma paralela controladas por un nodo *master*, el cual rige y administra todos los procesos que se ejecuta en los nodos trabajadores. Esta arquitectura toma en cuenta que en caso de fallo de alguno de los nodos trabajadores, el nodo *master* a través de Docker Swarm brinde los mecanismos necesarios para monitorear la disponibilidad de cada nodo trabajador. En el caso de la herramienta Spark, el nodo *master* es capaz de monitorear cada proceso de minado de datos que se realizan en los nodos trabajadores, así como el estado de disponibilidad de los mismos.

La utilización de los mecanismos antes descritos para recuperar información que está contenida dentro de los nodos trabajadores, permite que no exista una falla distribuida, haciendo posible a través de estos mecanismos que el sistema se recupere posteriormente de la ocurrencia de una falla. Docker Swarm brinda mecanismos para identificar las fallas que pueden ocurrir y actuar de forma inmediata para enmascararlos a través del nodo *master*, el cual se ocupa de realizar las copias pertinentes para que la información almacenada pueda ser recuperada de forma transparente.

2.5.2 Escalabilidad

La escalabilidad viene dada por la capacidad que presenta el sistema propuesto para crecer sin aumentar su complejidad, teniendo en cuenta que el mismo no disminuya su rendimiento. Mediante la utilización de Docker, cualquier servicio que maneje una carga adicional al aumentar el número de contenedores se le considera escalable horizontalmente.

Existen dos formas fundamentales de implementación al escalar un servicio en Docker:

- ✓ Modo paralelo (predeterminado): Todos los contenedores de un servicio se implementan al mismo tiempo sin existir ningún vínculo entre ellos. Esta forma es la más rápida de implementar además de ejecutarse de forma predeterminada.
- ✓ Modo secuencial: Cada contenedor nuevo se implementa en el servicio de uno por vez, cada contenedor se encuentra vinculado a todos los contenedores anteriores utilizando enlaces de servicio. Esto hace posible que la configuración se realice de forma más compleja dentro de la lógica de inicio de contenedores.

En el entorno desarrollado se aplicó de forma predeterminada el modo paralelo, debido a que en cada nodo trabajador se desarrollará un algoritmo diferente de minado de datos sin necesidad de que exista una relación entre ellos que no sea supervisada por el nodo *master*. Cuando los contenedores que son ejecutados en cada nodo, funcionan de forma independiente el uno del otro y no necesitan coordinarse entre sí, el modo paralelo puede ser ampliado, o sea, que pueden ser instaladas varios nodos trabajadores ejecutándose de forma independiente a través de un contenedor con una imagen de Spark para minar los datos.

El entorno desarrollado es escalable teniendo en cuenta las principales aristas por la cual se define este concepto, ya sea a través de su tamaño, ya que pueden ser incorporados nuevos contenedores

de trabajo en caso de que se desee incorporar nuevos servicios para el proceso de minería de datos. Otra característica fundamental que hace el entorno escalable es el acceso remoto, donde el *master* puede acceder a los nodos trabajadores propiciando la escalabilidad desde el punto de vista administrativo ya que monitorea constantemente el estado de cada uno de sus nodos distribuidos.

Por su parte Spark, se considera escalable debido a la capacidad de procesamiento de datos que utiliza en los nodos trabajadores. A través de la utilización de esta herramienta se pueden añadir más nodos al entorno desarrollado en dependencia de la necesidad de los algoritmos utilizados, para a partir de que los datos crezcan se puedan añadir más capacidad de cómputo.

2.5.3 Fiabilidad y tolerancia a fallos

La fiabilidad y la tolerancia a fallos son características a tomar en cuenta en el entorno desarrollado, ya que en cada nodo trabajador se ejecutarán algoritmos de minado de datos que serán capaces de identificar y extraer conocimiento de datos heterogéneos. En este sentido, el nodo *master* archiva toda la información que es producida por cada nodo trabajador respectivamente.

El nodo *master* debe respaldar todos los procesos que sean ejecutados en los nodos trabajadores, de manera que si existe una falla que pueda afectar el correcto funcionamiento del entorno, el mismo debe informar sobre la falla existente y debe haber realizado las réplicas pertinentes. En el caso de que el nodo *master* experimente una falla de sistema, no existe dentro de la arquitectura física un nodo que respalde todos los datos registrados en el mismo. Teniendo esta debilidad y retomando la arquitectura propuesta del funcionamiento de tres nodos trabajadores y un nodo *master*, se hace necesario reajustar la arquitectura planteada, ya que una falla de sistema de este tipo implica la afectación directa de todos los componentes del entorno.

Un sistema tolerante a fallos es aquel que a pesar de existir fallas en los procesos debe ser capaz de ocultar las mismas y seguir ejecutando los procesos sin interrupción. La técnica identificada para lograr un entorno tolerante a fallos es a través de la redundancia, la misma será aplicada para crear redundancia tanto en los nodos físicos como en la información que es producida en cada uno de ellos.

Cuando Docker Engine es ejecutado en modo Swarm, los nodos *master* implementan el algoritmo Raft para administrar el estado actual del *cluster* en su forma global. El método es utilizado para que los nodos *masters* mantengan un estado interno consistente de todo el entorno y de los servicios que se ejecutan en él. Obtener el mismo estado consistente en todo el *cluster* trae consigo que en caso de que el sistema experimente una falla, cualquier nodo *master* que se encuentre disponible puede obtener las tareas para restaurar los servicios a un estado de estabilidad y de disponibilidad de forma transparente al usuario y a las aplicaciones (Fabrizio Soppelsa, 2016).

Es recomendable para esta implementación ejecutar más de un nodo *master*, en caso de que uno de ellos falle, el otro nodo respalda la administración para que los servicios continúen ejecutándose. Para ello, debe mantenerse un número impar de administradores para soportar las fallas que puedan ocurrir en los mismos, ya que un número impar garantiza que durante una partición de red exista más posibilidad de que un nodo *master* permanezca disponible para procesar solicitudes si la red se divide en dos conjuntos (Fabrizio Soppelsa, 2016).

La aplicación del algoritmo Raft se implementa a través del cálculo de la cantidad de administradores (N), el cual tolera una pérdida máxima de N-1 nodos *master* entre la cantidad de nodos disponibles. Teniendo en cuenta que se desea concebir un entorno pequeño, y que la implementación de este mecanismo requiere un número impar de administradores, se aplica la siguiente fórmula (Fabrizio Soppelsa, 2016):

Partiendo de (N-1) /2, aplicando un total de tres nodos *master*, donde N es igual a 3, es decir (3-1), entre 2, que representa la cantidad de nodos que quedarán disponibles, se obtiene una tolerancia de fallos de una unidad.

Los nodos trabajadores se ejecutan como instancias de Docker Engine, teniendo como único propósito la ejecución de contenedores. Es importante aclarar que dichos nodos no participan en el estado distribuido de Raft, el cual es aplicado solamente en los nodos *masters*. La ejecución del modo Swarm puede estar dada a partir de un nodo *master*, pero no puede ejecutarse un nodo trabajador sin la ejecución de un *master*, donde por defecto todos estos nodos *masters* también se desarrollan como nodos trabajadores.

A través de la figura 43 que se presenta a continuación, se establece una arquitectura para el funcionamiento del entorno distribuido capaz de ser tolerante a los fallos que puedan ocurrir.

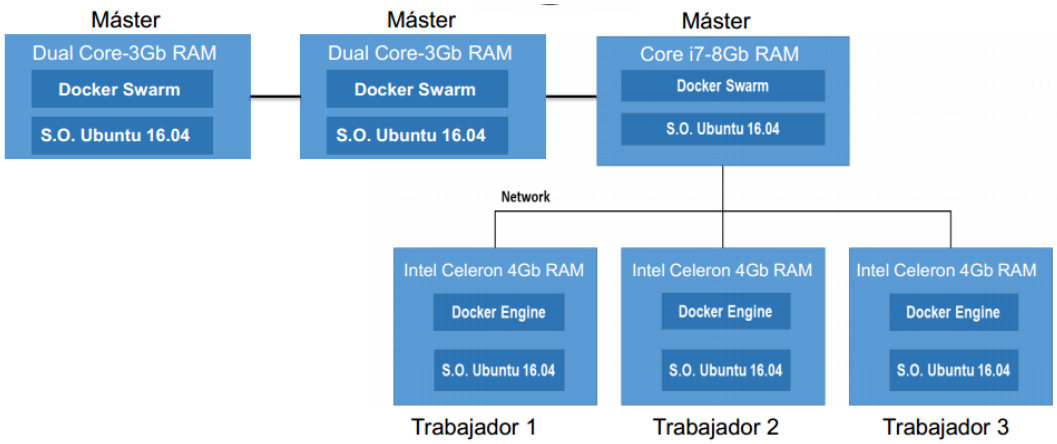


Figura 43: Aplicación de métodos tolerantes a fallas para el nodo master. Fuente: Elaboración propia.

Como se muestra, se aplicaron un total de tres *masters* a través de la implementación de Raft, los cuales garantizan la disponibilidad de los servicios que son ejecutados en cada nodo trabajador. Al fallar uno de los nodos que funciona como *master*, el que se encuentre disponible toma el control sobre los nodos trabajadores. De esta manera se asegura que al existir una falla parcial, el sistema puede recuperarse automáticamente sin afectar el funcionamiento de los servicios ni disminuir el rendimiento del mismo, ejecutando satisfactoriamente todas las funciones para las que ha sido diseñado el entorno distribuido.

La herramienta Spark por su parte, se considera tolerante a fallos, ya que la misma maneja automáticamente el fallo de un nodo trabajador. Spark provee a través de sus APIS la creación de algoritmos que procesan datos de forma distribuida, abstrayendo al programador de la tolerancia a fallos y la escalabilidad de su solución. Esto se hace posible a través de los RDD, que se definen como mecanismos de almacenamiento de *datasets* distribuidos al que se le pueden aplicar acciones

y trasformaciones que son ejecutadas en los nodos trabajadores. Los *datasets* distribuidos permiten la recuperación de información cuando se realiza una tarea en tiempo de ejecución, en caso de que ocurran perdidas de datos debido a fallos en la red.

Otro enfoque para propiciar que el entorno sea tolerante a los fallos que puedan ocurrir, está basado en las variables de difusión o *broadcast*, las cuales permiten que el trabajo con Spark conserve una copia de solo lectura de una variable en caché en cada nodo, en lugar de enviar una copia con cada tarea.

Mediante la aplicación de las diferentes técnicas analizadas, es posible garantizar un entorno capaz de ser tolerante a fallas, y a la vez capaz de propiciar la fiabilidad mediante su utilización. Como se evidenció anteriormente el manejo del método Raft en los nodos *masters* identificados, posibilita que se conserve un estado interno consistente en todo su conjunto, además de los servicios que son ejecutados, dando paso a través de su utilización que se obtenga un entorno capaz de ser tolerante a fallos. Spark por su parte, implementa varios mecanismos que facilitan la escalabilidad y tolerancia a fallos, de manera que durante el procesamiento si algún nodo rebajador falla, se realiza una recuperación de los datos de forma automática, recalculando el tamaño de los bloques que se distribuyen y volviendo a ejecutar el procedimiento en los nodos que quedan disponibles.

Conclusiones del capítulo:

- ✓ Se estableció el flujo general del entorno distribuido para el minado de datos en ambientes educacionales masivos, conformado por los procesos de gestión de datos, procesamiento de datos y virtualización del entorno.
- ✓ El proceso de gestión de datos y sus subprocesos asociados permiten aglutinar diversas bases de datos para su fácil acceso, contribuyendo a la escalabilidad del entorno.
- ✓ El flujo de procesamiento de datos permite la ejecución en paralelo de los algoritmos de minería de datos de forma transparente, lo cual garantiza la disponibilidad y la eficiencia dentro del entorno.
- ✓ El proceso de virtualización garantiza que el procesamiento de datos logre una adecuada disponibilidad y tolerancia a fallos, a través de la administración del *cluster* mediante la utilización de Docker Swarm.
- ✓ La arquitectura propuesta permite dar cumplimiento a las características de transparencia, escalabilidad y tolerancia a fallos, además de garantizar la eficiencia dentro del entorno distribuido para el minado de datos en ambientes educacionales masivos.

3 ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

El capítulo que se presenta a continuación, se procede a realizar la validación del entorno distribuido para el minado de datos educacionales masivos, teniendo en cuenta los métodos científicos teóricos y empíricos que fueron definidos en el marco teórico de la presente investigación. Para la realización de dicha validación es seleccionada la muestra. A partir de la misma, se aplica el método cualitativo criterio de expertos, la técnica de Iadov para medir la satisfacción del usuario y se aplica un experimento para comprobar la eficiencia de la propuesta de solución.

3.1 Descripción de la muestra utilizada

Para la realización del proceso de validación y aplicación de los métodos correspondientes antes mencionados, se tuvo en cuenta como escenario de ejecución de dichas pruebas, la Universidad de las Ciencias Informáticas. Se selecciona dentro de la universidad el área del departamento de programación correspondiente a la Facultad 4. Esta área se encuentra inmersa en la ejecución y puesta en marcha del proyecto de minado de datos educativos, correspondiente al período del mes de mayo-junio del año 2018.

Para llevar a cabo el proceso de pruebas se obtuvo una muestra de 15 personas. Los mismos se identifican como el 100% de su totalidad ingenieros en ciencias informáticas, un 96% de los mismos vinculado a proyectos de investigación y un 100% vinculado a la docencia.

Para el experimento realizado se tuvo en cuenta dos ambientes de ejecución para el minado de datos. El primero de ellos, a partir de un ***dataset*** con datos educacionales, realiza el proceso de minería de datos dentro del entorno distribuido desarrollado, mientras que el segundo lo realiza fuera del mismo.

3.2 Aplicación de la técnica de la técnica de Iadov para medir la satisfacción del usuario

La técnica de Iadov creada para establecer el nivel de satisfacción por la profesión de carreras pedagógicas. No obstante, ha sido modificada por algunos autores para su aplicación en la valoración de la satisfacción en disímiles campos de estudio, como parte de diagnósticos y validaciones en múltiples investigaciones (Cedeño, 2013) (Díaz, 2012).

La utilización de esta técnica está basada en la aplicación y el análisis de un cuestionario con una estructura interna determinada, que establece una relación entre tres preguntas cerradas y la realización de un análisis posterior de otro conjunto de preguntas abiertas (Cedeño, 2013) (Díaz, 2012).

La tabla que se muestra a continuación muestra la relación entre las preguntas cerradas que se establecen a través del Cuadro Lógico de Iadov, el cual posibilita con su utilización estimar el nivel de satisfacción de los usuarios.

Tabla 1: Cuadro Lógico de Iadov. Fuente: Elaboración propia.

	¿Considera usted importante el desarrollo de un entorno distribuido que contribuya al minado de datos en ambientes educacionales masivos?								
	Si			No			No sé		
	¿Si usted necesitara ejecutar procesos de minería de datos educacionales o facilitar su trabajo en este ámbito de forma eficiente, utilizaría este entorno informático desarrollado?								
¿Le satisface la composición del entorno para el minado de datos en ambientes educacionales masivos?	Si	No sé	No	Si	No sé	No	Si	No sé	No
Me satisface mucho	1	2	6	2	2	6	6	6	6
Más satisfecho que insatisfecho	2	2	3	2	3	3	6	3	6
Me es indiferente	3	3	3	3	3	3	3	3	3
Me insatisface más de lo que me satisface	6	3	6	3	4	4	3	4	4
No me satisface nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

La autora de la presente investigación, aplicó la técnica antes mencionada para medir la satisfacción de los usuarios con respecto al entorno distribuido para el minado de datos en ambientes educacionales masivos. Los resultados obtenidos están basados en la escala de satisfacción que indica la técnica empleada, además de la aplicación de la fórmula para determinar el Índice de Satisfacción Grupal (ISG). La escala de satisfacción responde a la estructura que se muestra a continuación, donde el número resultante de la interrelación de las tres preguntas cerradas indica la posición de cada evaluador:

1. Clara satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho que satisfecho.
5. Clara insatisfacción.
6. Contradictoria.

Para la realización de la ponderación del ISG, es establecida una escala numérica entre +1 y -1, de la forma que se muestra:

- ✓ +1 máximo de satisfacción.
- ✓ +0.5 más satisfecho que insatisfecho.
- ✓ 0 indefinido y contradictorio.
- ✓ -0.5 más insatisfecho que satisfecho.

- ✓ -1 máxima insatisfacción.

El cálculo del Índice de Satisfacción Grupal (ISG) está dado por la fórmula que se muestra a continuación:

$$\text{ISG}= \frac{\text{A}(+1)+\text{B}(+0{,}5)+\text{C}(0)+\text{D}(-0{,}5)+\text{E}(-1)}{\text{N}}$$

Donde se permite conocer las categorías grupales siguientes:

- ✓ Insatisfacción: desde (-1) hasta (-0,5).
- ✓ Contradictorio: desde (-0,49) hasta (+0,49).
- ✓ Satisfacción: desde (+0,5) hasta (1).

Para la determinación del grado de satisfacción se tuvo en cuenta un total de 15 personas como fue citado anteriormente. Los mismos fueron escogidos por factores tales como los años de experiencia en el campo informático, la categoría docente y científica que ocupa, así como el perfil en el cual se desarrolla. Dichos datos son mostrados en la siguiente tabla:

Tabla 2: Relación de personas implicadas en la validación. Fuente: Elaboración propia.

	Descripción	Cantidad
Perfil en el que se desarrolla	Directivo	5
	Especialista	0
	Profesor	10
Años de experiencia	Hasta 5 años	8
	De 5 hasta 10 años	6
	Más de 10 años	1
Categoría docente	Titular	0
	Auxiliar	3
	Asistente	3
	Instructor	9
	Ninguna	
Categoría científica	Doctor	0
	Máster	2
	Ninguna	13

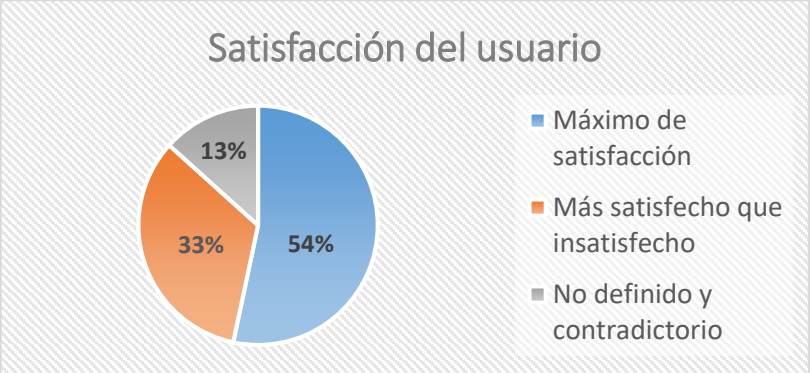
A partir de esta selección, se aplicó un cuestionario de seis preguntas (Ver Anexo 1) para a través de las respuestas obtenidas determinar el grado de satisfacción del usuario encuestado. Sustituyendo en la ecuación de ISG se determina lo siguiente:

$$\text{ISG}= \frac{8(+1)+5(+0{,}5)+2(0)+0(-0{,}5)+0(-1)}{15}$$

$$\text{ISG}= \mathbf{0{,}966}$$

El valor obtenido al aplicar la técnica es de 0,966 cuyo resultado se encuentra situado en el intervalo de satisfacción, lo que demuestra que existe un alto grado de satisfacción con el entorno desarrollado. A continuación se expone un gráfico que demuestra los resultados obtenidos.

Tabla 3: Porciento de satisfacción de los usuarios con el entorno. Fuente: Elaboración propia.



El cuestionario aplicado también contó con preguntas abiertas, las cuales posibilitaron profundizar en aspectos positivos y negativos que conforman el entorno desarrollado, importantes para el perfeccionamiento del mismo. Entre los aspectos que se enunciaron como positivos se encuentran los siguientes:

- ✓ Propicia un ambiente factible para el análisis de grandes volúmenes de datos educacionales.
- ✓ Facilita el análisis de grandes volúmenes de información a gran velocidad y permite la utilización de hardware que no necesita de grandes prestaciones.
- ✓ Proporciona escalabilidad y brinda un entorno de trabajo transparente al usuario.

Sobre los elementos negativos abordados fueron de mínima relevancia, los cuales fueron analizados y corregidos instantáneamente.

3.3 Aplicación del método criterio de expertos

La aplicación del criterio de expertos, está basado en la obtención de valoraciones de expertos relacionadas con temas de la propuesta de solución. Como método para el procesamiento estadístico de estos criterios, fue aplicada la escala psicométrica creada por Rensis Likert en 1932 (Likert, 1932). La misma es utilizada para determinar a través de un cuestionario el nivel de acuerdo o desacuerdo de la conformación del entorno distribuido desarrollado, a través del procesamiento estadístico de los criterios y evaluaciones expresadas.

Los indicadores que fueron seleccionados para ser evaluados por los expertos, pueden ser consultados en el Anexo 2.

En cuanto a la selección de los expertos, se tuvo en cuenta los años de experiencia en el campo, la categoría científica, el área de trabajo en que se desarrolla y el nivel de dominio que presenta respecto al tema. Los candidatos a expertos realizaron una encuesta (Ver Anexo 3) para determinar el coeficiente de competencia de cada uno, lo que asegura la confiabilidad de las respuestas que fueron ofrecidas.

El nivel de competencia de cada experto, se determinó como parte del aseguramiento de la confiabilidad de las respuestas mediante el cálculo de su coeficiente de competencia. Para determinar el coeficiente de competencia, fue seguido el procedimiento que es mostrado en el

Anexo 4, mientras que los resultados que fueron arrojados luego de aplicada la encuesta pueden ser consultados en el Anexo 5. La distribución de los expertos según su nivel de competencia se pueden observar a continuación:

Tabla 4: Distribución de los expertos según el nivel de competencia. Fuente: Elaboración propia

Nivel de competencia	**Cantidad**	**Porciento**
Alto	8	67%
Medio	3	25%
Bajo	1	8%
Total	12	100%

Después de analizada la muestra del comportamiento de los niveles de competencia, se determinó seleccionar un total de **11** expertos, ya que el nivel de competencia que poseen es adecuado para los elementos teóricos que se desean analizar. Dicha cantidad, resulta apropiada para garantizar la confiabilidad de los resultados. La composición de la relación de los expertos analizados se describe en la siguiente tabla:

Tabla 5: Composición de la relación de expertos. Fuente: Elaboración propia.

	Descripción	**Cantidad**
Perfil en el que se desarrolla	Directivo	4
	Especialista	5
	Profesor	2
Años de experiencia	Hasta 5 años	3
	De 5 hasta 10 años	5
	Más de 10 años	3
Categoría docente	Titular	1
	Auxiliar	3
	Asistente	5
	Instructor	2
	Ninguna	0
Categoría científica	Doctor	2
	Máster	4
	Especialista	5
Total de expertos	11	

Las preguntas del cuestionario (Ver Anexo 2), están orientadas a la obtención de valoraciones de los expertos en cuestión, enfocadas a los indicadores y problemas que fueron definidos. Los indicadores a tener en cuenta se expresan mediante valoraciones que muestra la siguiente escala:

5: MUY DE ACUERDO (MA).

4: DE ACUERDO (DA).

3: NI DE ACUERDO NI EN DESACUERDO (Si-No).

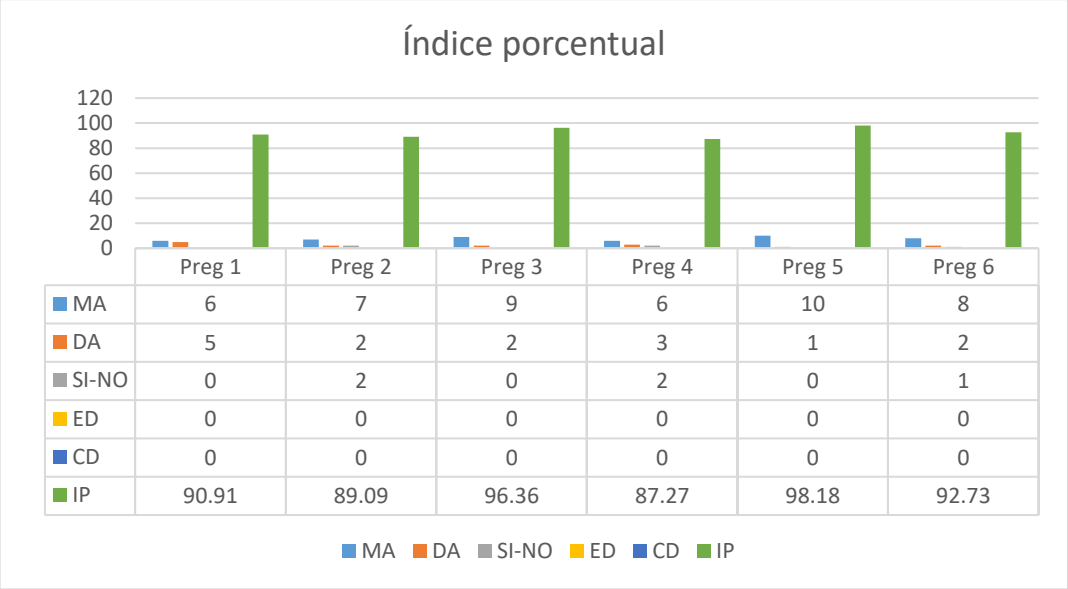
2: EN DESACUERDO (ED).

1: COMPLETAMENTE EN DESACUERDO (CD).

Posteriormente son expresados los resultados en la escala de Lickert. A través de esta técnica, son calculados los porcientos de concordancia de los expertos con cada una de las posibles respuestas para los planteamientos formulados (Ver Anexo 6). Posteriormente es calculado el índice porcentual (IP), el cual integra en un solo valor la aceptación de cada planteamiento por los evaluadores mediante la fórmula que se plantea a continuación:

$$IP = \frac{5\ (\%\ \text{de MA})\ +\ 4\ (\%\ \text{de DA})\ +\ 3\ (\%\ \text{de SI}-\text{NO})\ +\ 2\ (\%\ \text{de ED})\ +\ 1\ (\%\ \text{de CD})}{5}$$

La figura 44, indica el índice porcentual de cada valoración obtenida por cada experto con relación a los aspectos tratados con anterioridad.



	Preg 1	Preg 2	Preg 3	Preg 4	Preg 5	Preg 6
MA	6	7	9	6	10	8
DA	5	2	2	3	1	2
SI-NO	0	2	0	2	0	1
ED	0	0	0	0	0	0
CD	0	0	0	0	0	0
IP	90.91	89.09	96.36	87.27	98.18	92.73

Figura 44: Índice Porcentual de Concordancia de los expertos. Fuente: Elaboración propia.

Como se puede constatar, el índice porcentual obtenido en cada uno de los casos es superior a un **80%**, lo que evidencia una alta valoración expresada por los expertos con relación a los indicadores analizados.

3.4 Experimento para evaluar la eficiencia del entorno distribuido

Según V. R. Basili (Basili, 1996), se plantea que *"un experimento es un estudio que involucra la manipulación intencional de una acción para analizar sus posibles efectos, y se lleva a cabo para analizar si una o más variables independientes afectan a una o más variables dependientes y por qué la afectan"*.

Con el fin de manipular la variable independiente, entorno distribuido, en función de la variable dependiente, eficiencia, se lleva a cabo un experimento, que permitirá comprobar el cumplimiento de la hipótesis planteada en la presente investigación.

Para la realización de este método, fueron minados datos educacionales a partir de un *dataset* de prueba dentro y fuera del entorno distribuido desarrollado. Este tipo de experimento, permitió que a través de los resultados arrojados en ambos caso, se pudiera establecer un análisis y comparación de los mismos para evaluar la eficiencia en cada caso.

Con los datos obtenidos por cada variante, se aplicaron criterios estadísticos que permiten establecer una comparación. La muestra utilizada estuvo dada por un total de 20 iteraciones, de las cuales fueron tomadas 19 de las mismas para su análisis. El resultado de la iteración eliminada no se encontró sobre el rango de los datos obtenidos, ya que el mismo se obtuvo como consecuencia de fallos y desperfectos técnicos que ocurrieron en el hardware durante el proceso de pruebas efectuado. El *dataset* utilizado para la prueba contó con un tamaño de 200x50, el cual es representado en un vector de respuesta dicotómico de *X* filas con *Y* columnas. La variable X representa el número de estudiantes, mientras que *Y* es la cantidad de ítems evaluativos de la muestra.

El hardware utilizado para el proceso de prueba, contó con un total de tres nodos trabajadores y un nodo *master*, los cuales presentan las siguientes características:

Tabla 6: Propiedades de los nodos trabajadores. Fuente: Elaboración propia.

Nodos Trabajadores					
Tipo de procesador	**Velocidad**	**Cantidad de núcleos**	**Memoria RAM**	**Sistema Operativo**	**Versión del Núcleo**
Intel Celeron	2.8GH	2	4GB	Ubuntu 16-0.4 LTS 64 bit	4.4.0-121-Generic

Tabla 7: Propiedades del nodo master. Fuente: Elaboración propia.

Nodo Master					
Tipo de procesador	**Velocidad**	**Cantidad de núcleos**	**Memoria RAM**	**Sistema Operativo**	**Versión del Núcleo**
Core i7 -4790	3.6 GH	8	8GB	Ubuntu 16-0.4 LTS 64 bit	4.4.0-121-Generic

3.4.1 Prueba de ajuste de datos no censurados – fuera del entorno

Datos/Variable: FUERA ENTORNO

19 valores con rango desde 60226,0 a 62377,0

Distribuciones ajustadas

Tabla 8: Distribución normal para los datos Fuera del entorno. Fuente: Elaboración propia.

Normal
media = 60926,3
desviación estándar = 593,663

Este análisis muestra los resultados de ajustar una distribución normal a los datos de fuera del entorno.

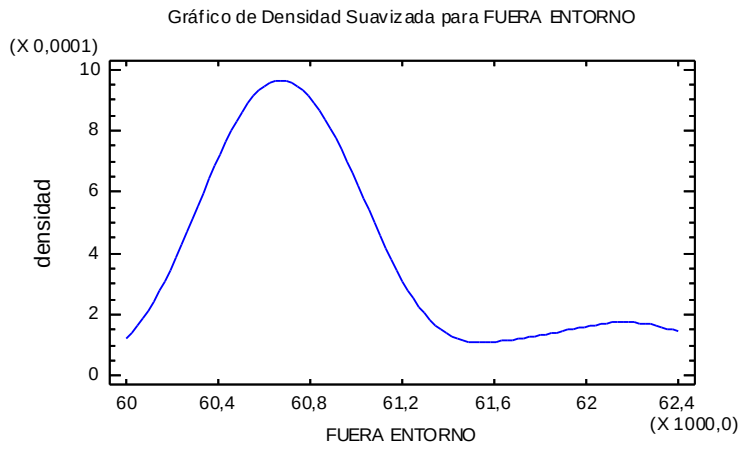


Figura 45: Gráfico de densidad suavizada para los datos Fuera del entorno. Fuente: Elaboración propia.

3.4.2 Prueba de ajuste de datos no censurados – dentro del entorno

Datos/Variable: DENTRO ENTORNO

19 valores con rango desde 41080,0 a 44238,0

Distribuciones Ajustadas

Tabla 9: Distribución normal para los datos Dentro del entorno. Fuente: Elaboración propia.

Normal
media = 41950,7
desviación estándar = 1029,77

Este análisis muestra los resultados de ajustar una distribución normal a los datos de dentro del entorno.

Luego de haber realizado el cálculo de la media para ambos casos se procede a realizar las pruebas de normalidad.

3.4.3 Pruebas de normalidad – fuera del entorno

Tabla 10: Pruebas de normalidad realizadas Fuera del entorno. Fuente: Elaboración propia.

Prueba	Estadístico	Valor-P
Chi-Cuadrado	27,1053	0,00134377
Estadístico W de Shapiro-Wilk	0,779897	0,000344473

La tabla muestra los resultados de diversas pruebas realizadas para determinar si fuera del entorno puede modelarse adecuadamente con una distribución normal. La prueba de chi-cuadrada divide el rango de la variable FUERA ENTORNO en 12 clases igualmente probables y compara el número de observaciones en cada clase con el número esperado de observaciones. La prueba de Shapiro-Wilk está basada en la comparación de los cuartiles de la distribución normal ajustada a los datos.

Debido a que el valor -P más pequeño de las pruebas realizadas es menor a **0,05**, se puede rechazar la idea de que la variable FUERA ENTORNO proviene de una distribución normal con **95%** de confianza.

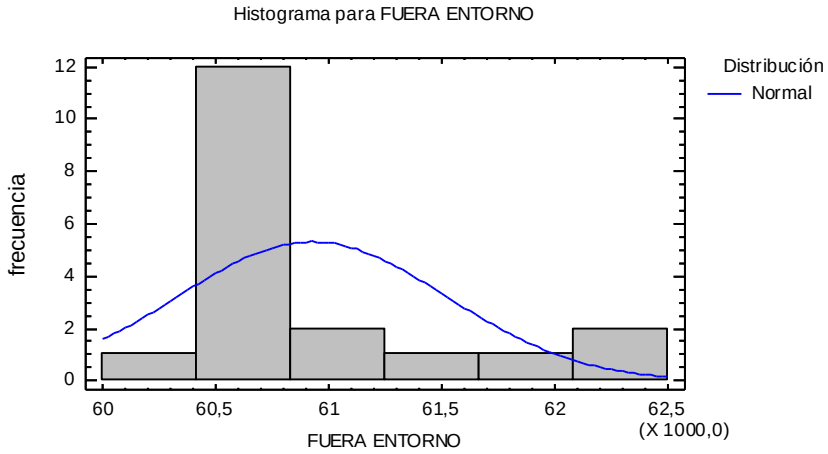


Figura 46: Histograma de prueba de normalidad para datos Fuera del entorno. Fuente: Elaboración propia.

3.4.4 Pruebas de normalidad – dentro del entorno

Tabla 11: Pruebas de normalidad realizadas Dentro del entorno. Fuente: Elaboración propia.

Prueba	Estadístico	Valor-P
Chi-Cuadrado	41,0	0,00000500194
Estadístico W de Shapiro-Wilk	0,687168	0,0000122205

La tabla muestra los resultados de diversas pruebas realizadas para determinar si dentro del entorno puede modelarse adecuadamente con una distribución normal. La prueba de chi-cuadrada divide el rango de la variable DENTRO ENTORNO en 12 clases igualmente probables y compara el número de observaciones en cada clase con el número esperado de observaciones. La prueba de Shapiro-Wilk está basada en la comparación de los cuartiles de la distribución normal ajustada a los datos.
Debido a que el valor -P más pequeño de las pruebas realizadas es menor a **0,05**, se puede rechazar la idea de que la variable DENTRO ENTORNO proviene de una distribución normal con **95%** de confianza.

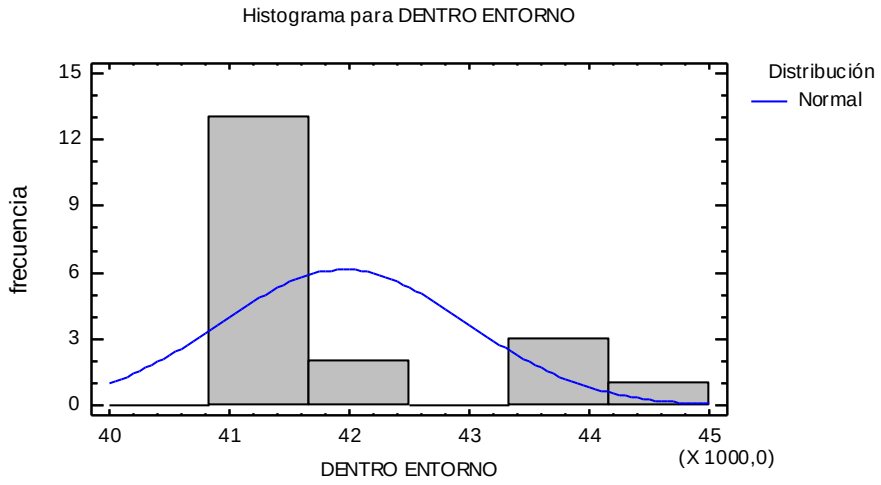


Figura 47: Histograma de prueba de normalidad para datos Dentro del entorno.

3.5 Resumen estadístico

Tabla 12: Resumen estadístico para las dos muestras de datos. Fuente: Elaboración propia.

	FUERA ENTORNO	DENTRO ENTORNO
Recuento	19	19
Promedio	60926,3	41950,7
Mediana	60769,0	41521,0
Varianza	352436	1,06043E6
Desviación Estándar	593,663	1029,77
Coeficiente de Variación	0,974396%	2,45472%
Mínimo	60226,0	41080,0
Máximo	62377,0	44238,0
Rango	2151,0	3158,0
Sesgo Estandarizado	2,796	2,69959
Curtosis Estandarizada	1,42096	0,605755

La tabla contiene el resumen estadístico para las dos muestras de datos. De particular interés son el sesgo estandarizado y la curtosis estandarizada que pueden usarse para comparar si las muestras provienen de distribuciones normales. Valores de estos estadísticos fuera del rango de -2 a +2 indican desviaciones significativas de la normalidad, lo que tendería a invalidar las pruebas que comparan las desviaciones estándar. En este caso, ambas muestras tienen valores de sesgo estandarizado fuera del rango normal. Ambas curtosis estandarizadas se encuentran dentro del rango esperado.

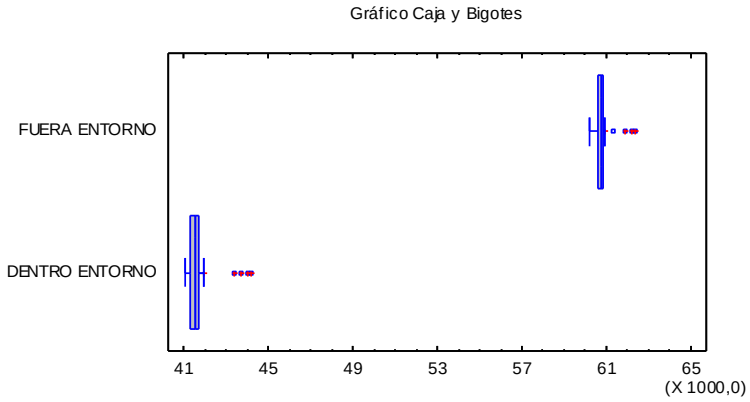


Figura 48: Comparación de las dos muestras utilizadas.

Intervalos de confianza del 95,0% para la media de la variable FUERA ENTORNO: 60926,3 +/- 286,137 [60640,1; 61212,4]

Intervalos de confianza del 95,0% para la media de la variable DENTRO ENTORNO: 41950,7 +/- 496,335 [41454,4; 42447,1]

Intervalos de confianza del 95,0% intervalo de confianza para la diferencia de medias sin suponer varianzas iguales: 18975,5 +/- 557,91 [18417,6; 19533,4]

Se ejecuta una prueba -t para comparar las medias de las dos muestras. También se construye los intervalos o cotas de confianza para cada media y para la diferencia entre las medias. De interés particular es el intervalo de confianza para la diferencia entre las medias, el cual se extiende desde 18417,6 hasta 19533,4. Puesto que el intervalo no contiene el valor 0, existe una diferencia estadísticamente significativa entre las medias de las dos muestras, con un nivel de confianza del 95,0%.

Prueba t para comparar medias:

Hipótesis nula: media1 = media2

Hipótesis Alt.: media1 <> media2

Sin suponer varianzas iguales: t = 69,5857 valor-P = 0,0

Se rechaza la hipótesis nula para alfa = 0,05.

La prueba que se presenta, se ha construido para determinar si la diferencia entre las dos medias es igual a 0,0 versus la hipótesis alterna de que la diferencia no es igual a 0,0. Puesto que el valor -P calculado es menor que 0,05, se puede rechazar la hipótesis nula en favor de la alterna.

3.5 Comparación de desviaciones estándar

Variables	FUERA ENTORNO	DENTRO ENTORNO
Desviación Estándar	593,663	1029,77
Varianza	352436,	1,06043E6
Gl	18	18

Razón de Varianzas= 0,332352

Intervalos de confianza del 95,0%

Desviación Estándar de la variable FUERA ENTORNO: [448,579; 877,923]

Desviación Estándar de la variable DENTRO ENTORNO: [778,109; 1522,85]

Razones de Varianzas: [0,128045; 0,86265]

Prueba -F para comparar Desviaciones Estándar

Hipótesis Nula: sigma1 = sigma2

Hipótesis Alt.: sigma1 <> sigma2

F = 0,332352 valor-P = 0,0244025

Se rechaza la hipótesis nula para alfa = 0,05.

Se ejecuta una prueba -F para comparar las varianzas de las dos muestras. También construye intervalos o cotas de confianza para cada desviación estándar y para la razón de varianzas. De particular interés es el intervalo de confianza para la razón de varianzas, el cual se extiende desde 0,128045 hasta 0,86265. Puesto que el intervalo no contiene el valor de 1, existe diferencia estadísticamente significativa entre las desviaciones estándar de las dos muestras con un 95,0%.

3.6 Comparación de medianas

Se ejecuta la prueba W de Mann-Whitney, prueba no paramétrica para comparar las medianas de las dos muestras. Esta prueba se construye combinando las dos muestras, ordenando los datos de menor a mayor, y comparando los rankeos promedio de las dos muestras en los datos combinados. Debido a que el valor -P es menor que 0,05, existe una diferencia estadísticamente significativa entre las medianas con un nivel de confianza del 95,0%.

Mediana de muestra 1: 60769,0

Mediana de muestra 2: 41521,0

Prueba W de Mann-Whitney (Wilcoxon) para comparar medianas:

Hipótesis Nula: mediana1 = mediana2

Hipótesis Alt.: mediana1 <> mediana2

Rango Promedio de la muestra 1: 29,0

Rango Promedio de la muestra 2: 10,0

W = 0,0 valor -P = 1,48277E-7

Se rechaza la hipótesis nula para alfa = 0,05, lo que significa que las medinas difieren significativamente.

Teniendo que:

Intervalos de confianza del 95,0% para la media de la variable FUERA ENTORNO: 60926,3 +/- 286,137 [60640,1; 61212,4]

Intervalos de confianza del 95,0% para la media de la variable DENTRO ENTORNO: 41950,7 +/- 496,335 [41454,4; 42447,1]

Demostrándose que el intervalo de confianza fuera del entorno oscila entre los sesenta a los sesenta y un mil, y que dentro del entorno oscila entre los cuarenta y un mil a los cuarenta y dos mil, se puede concluir que dentro del entorno la eficiencia dada por la velocidad de procesamiento de los datos minados, es 1/3 más eficiente que los datos que son minados fuera del entorno.

Conclusiones del capítulo:

- ✓ A través de la aplicación de la técnica de Iadov para estimar la satisfacción del usuario se pudo constatar la satisfacción del usuario con el entorno distribuido para el minado de datos desarrollado.
- ✓ La aplicación del método criterio de expertos mediante la escala de Lickert, demostró que los fundamentos que basan la investigación son veraces, obteniéndose en todos los indicadores analizados un porciento de concordancia superior al 80%.
- ✓ El experimento realizado permitió comprobar la eficiencia en cuanto a tiempo de respuesta de los datos dentro del entorno distribuido desarrollado, a través de la prueba Shapiro – Wilk y la prueba no paramétrica de Mann – Whithey.
- ✓ El entono distribuido para el minado de datos en ambientes educacionales masivos presentado contribuye al uso de Técnicas de Minería de Datos Distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos, lo cual corrobora la hipótesis planteada al inicio de la investigación.

CONCLUSIONES GENERALES

Finalizada la investigación se arriban a las siguientes conclusiones generales:

- ✓ El análisis de conceptos y tecnologías, permitió identificar las características y seleccionar las herramientas necesarias para el desarrollo del entorno distribuido para el minado de datos en ambientes educacionales masivos.
- ✓ Los procesos de gestión de datos, procesamiento de datos y virtualización del entorno, asociados a la arquitectura propuesta, permiten dar cumplimiento a las características de transparencia, escalabilidad y tolerancia a fallos, y garantizan la eficiencia dentro del entorno propuesto.
- ✓ Los métodos científicos utilizados para la validación demuestran la eficiencia del entorno y su factibilidad para ser utilizado en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos.

RECOMENDACIONES

- ✓ Desarrollar una aplicación web para la configuración y administración de los procesos de minado de datos y para los del *cluster*.
- ✓ Desarrollar una arquitectura donde cada algoritmo de estimación de conocimiento latente se ejecute como un micro servicio para ser ejecutado de forma independiente en un contenedor Docker.
- ✓ Desarrollar un sistema multiagente que tome como base el entorno distribuido para el minado de datos en ambientes educacionales masivos.

BIBLIOGRAFÍA

Aggarwal, C. C. (2015). *Data Mining: The Textbook.* Springer. doi:10.1007/978-3-319-14142-8

Allae Erraissi, A. B. (Diciembre de 2017). A Comparative Study of Hadoop-based Big Data Architectures. *International Journal of Web Applications, 9*(4), 129-137.

Andrew S. Tanenbaum, M. V. (2007). *Distributed Systems: principles and paradigms.* New Jersey: Pearson Prentice Hall .

B.Thillaieswari M.S., M. B. (2017). Comparative Study on Tools and Techniques of Big Data Analysis. *International Journal of Advanced Networking & Applications (IJANA), Vol: 8*(Issue: 05), 61-66.

Baker, R. S. (2011). Data Mining for Education. *International Encyclopedia of Education, 3*.

Baker, R. S. (2014). Educational Data Mining and Learning Analytics. *The Cambridge Handbook of the Learning Sciences*, 253-272. doi:10.1017/CBO9781139519526.016

Ballesteros Román, A., & Daniel Sánchez-Guzmán, R. G. (2014). Minería de datos educativa: Una herramienta para la investigación de patrones de aprendizaje sobre un contexto educativo. *7*(4), 662-668.

Basili, V. R. (1996). The role of experimentation: past, present, future (keynote presentation). Berlin, Germany: 18th International Conference on Software Engineering.

Carlos Márquez Vera, C. R. (2012). Predicción del Fracaso Escolar mediante Técnicas de Minería de Datos. *Vol. 7*(Núm. 3).

Cedeño, D. M. (2013). *Ambiente de trabajo para la producción de objeos de aprendizaje en la educación superior.* Centro de tecnologías para la formación. La Habana: Universidad de las Ciencias Informáticas.

Chikhale, R. (2016). Study of Distributed Data Mining Algorithm and Trends. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 41-47.

Chuck, L. (2011). *Hadoop in Action.* Manning Publications Co.

Constanza R. Huapaya, F. A. (2012). Minería de Datos Educacional en Ambientes. (223-4816600 int 259).

Corbett, R. S. (2014). Assessment of robust learning with educational data mining. *9*.

Cristobal Romero, S. V. (2013). Data mining in education. *WIREs Data Mining and knowledge discovery, 3*, 12-27. doi: 10.1002/widm.1075

D. Talia, P. T. (2006). Grid-Based Distributed Data Mining Systems, Algorithms and Services. *In Proceedings of the 9th International Workshop on High Performance and Distributed Mining (HPDM).* Portugal: Springer.

Deepika P, A. R. (Septiembre de 2015). A Study of Hadoop-Related Tools and Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering, 5*(Issue 9), 160-164.

Devi, S. G. (Marzo de 2014). A Survey on Distributed Data Mining and ITS Trends. *IMPACT: International Journal of Research in Engineering & Technology (IMPACT: IJRET), 2*(Issue 3), 107-120.

Díaz, O. F. (2012). *MIDAC: Modelo para el desarrollo de aplicaciones compuestas.* Facultad 5. La Habana: Universidad de las Ciencias Informáticas.

Diego Martín, M. M.-A. (2011). Virtualización, una solución para la eficiencia, seguridad y administración de intranets. *20*(3).

Diliu, Z. L. (2014). The Research and Implementation of Cloud Computing Platform based on Docker.

Dillip Kumar Swain, S. M. (Mayo de 2017). Privacy Preservation in Distributed Data Mining Using Secured Multiparty Computation . *International Journal of Scientific Research Engineering & Technology (IJSRET), 6*(Issue 5), 494-499.

Dubitzky, W. (2008). *Data Mining Techniques in Grid Computing Environments.* UK: John Wiley & Sons.

E. E. Millán-Rojas, A. P.-T.-V. (2016). Simulación de una red Grid con máquinas virtuales para crear un entorno de aprendizaje de la computación de alto desempeño. *25* (41).

Fabrizio Soppelsa, C. K. (2016). *Native Docker Clustering with Swarm.* Packt Publishing.

Fúquene, J. A. (2017). *Diseño y aportes de un modelo para minería de datos educativos en aulas de educación media de carácter presencial.* Tesis doctoral, Universidad de Santo Tomás, Facultad de educación, Bogotá.

George Coulouris, J. D. (2012). *Distributed Systems Concepts and Design* . London: Addison-Wesley.

Guller, M. (2015). *Big Data Analytics with Spark A Practitioner's Guide to Using Spark for Large Scale Data Analysis.* Apress.

Jain, R. (Agosto de 2015). Introduction to Data Mining Techniques. Obtenido de https://www.researchgate.net/publication/265151471_Introduction_to_Data_Mining_Techniques

James E. Smith, R. N. (2005). The Architecture of Virtual Machines.

Javier Rey, M. C. (2015). Efficient Prototyping of Fault Tolerant Map-Reduce Applications with Docker-Hadoop. *International Conference on Cloud Engineering IEEE*.

Jesús A. Castorena Peña, A. E. (Junio de 2018). El uso de herramientas tecnológicas de minería de datos en el análisis de datos climatológicos. *Revista Iberoamericana de las Ciencias Computacionales e Informática (RECI), 7*(13), 1-18. doi:10.23913/reci.v7i13.75

Jiawei Han, M. K. (2006). *Data Mining: Concepts and Techniques* (Segunda ed.). San Francisco: Elsevier.

Johann Ari Larusson, B. W. (2014). Learning Analytics.

Judy Kay, I. K. (2011). Educational Data Mining to Support Group Work in Software Development Projects. En S. V. Cristóbal Romero, *Handbook of Educational Data Mining* (págs. 173-184). CRC Press Taylor & Francis Group.

Kavitha, G., & Raj, L. (Marzo de 2017). Educational Data Mining and Analytics - Educational Assistance for Teaching and Learning. *International Journal of Computer & Organization Trends (IJCOT), 41*(1).

Lafuente, A. (2015). Introducción a los sistemas distribuidos. En A. Lafuente. UPV/EHU .

Lam, C. (2011). *Hadoop in action.* Stamford: Manning Publications Co.

Lambodar Jena, N. K. (Noviembre de 2015). Distributed Data Mining Classification Algorithms for Prediction of Chronic- Kidney-Disease. *International Journal of Emerging Research in Management &Technology, 4*(Issue-11), 110-118.

Laura Calvet Liñán, Á. A. (2015). Educational Data Mining and Learning Analytics: differences, similarities, and time evolution. *Universities and Knowledge Society Journal,*, 98-112.

Likert, R. (1932). *A technique for the measurement of attitudes. Archives of psychology.* Pac, S.A. de C.V.

Mafruz Zaman Ashrafi, D. T. (2002). *A Data Mining Architecture for Clustered Environments.* Finland: Springer-Verlag .

Maria A. Murazzo, N. R. (2016). Identificación de Algoritmos de Cómputo Intensivo para Big Data y su Implementación en Clouds. *Workshop de Investigadores en Ciencias de la Computación (WICC)* (págs. 767-771). Argentina: XVIII Workshop de Investigadores en Ciencias de la Computación.

Martínez, J. J. (2015). Análisis de minería de datos distribuida con Weka Parallel en computadoras con múltiples procesadores físicos y lógicos. *Economía y Administración (E&A), 6*(2), 154-165.

Maya-Pérez, P. N.-C.-R.-A. (Junio de 2016). Propuesta del diseño de un Modelo Predictivo de alerta temprana en indicadores educativos de Nivel Superior aplicando Minería de datos. *Revista de Aplicación Científica y Técnica, 2*(4), 29-40.

Mr. Piyush Bhardwaj, A. G. (Mayo de 2016). A Survey on Comparative Analysis of Big Data Tools. *International Journal of Computer Science and Mobile Computing, 5*(Issue. 5), 789 – 793.

Nabila Bousbia, I. B. (2014). Which Contribution Does EDM Provide to Computer-Based Learning Environments? En A. Peña-Ayala, *Educational Data Mining Applications and Trends* (págs. 3-28). Springer . doi:10.1007/978-3-319-02738-8_1

Néstor D. Duque Méndez, E. J. (2016). Modelo para el proceso de extracción, transformación y carga en bodegas de datos. Una aplicación con datos ambientales. *Ciencia e Ingeniería Neogranadina, 26*, 95-10. doi:https://doi.org/10.18359/rcin.1799

Nickoloff, J. (2016). *Docker in Action.* Shelter Island: Manning Publications Co.

Nikyle Nguyen, D. B. (2017). Distributed MPI cluster with Docker Swarm mode. Las Vegas: IEEE. doi:10.1109/CCWC.2017.7868429

Ocampo, M. G. (2017). *Descubrimiento de patrones en interacciones entre estudiantes y plataformas virtuales de educación mediante el uso de analíticas de aprendizaje.* Tesis de maestría, Universidad Nacional de Colombia, Departamento de Ciencias de la Computación y de la Decisión, Medellín.

Perlich, C. (Enero de 2011). Learning Curves in Machine Learning. *IBM Research Report*. doi:10.1007/978-0-387-30164-8_452

Prachi Pandey, S. S. (2016). A Comparative Study of Hadoop Family Tools. *(IJCSIT) International Journal of Computer Science and Information Technologies, 7*(3), 1620-1623.

R. Sampieri, C. F. (2013). *Metodología de la Investigación.* México D. F: Mc Graw-Hill.

R.Vijayakumari, R. K. (Febrero de 2014). Comparative analysis of Google File System and Hadoop Distributed File System. *International Journal of Advanced Trends in Computer Science and Engineering, 3*(1), 553– 558.

Ramageri, B. M. (2010). Data Mining Techniques and applications. *Indian Journal of Computer Science and Engineering, 1*(4), 301-305.

Rodríguez, P. M. (2015). *Análisis de sinergias en despliegue de sistemas de computación distribuida: APACHE SPARK-OPENSTACK.* Tesis de maestría, Universidad Politécnica de Madrid Escuela Técnica Superior de Ingenieros de Telecomunicación.

Rodríguez, Z. E. (2015). *Aplicación de la minería de datos distribuida usando algoritmo de clustering K-MEANS para mejorar la calidad de servicios de las organizaciones modernas.* Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Cinecias Mateméticas, Lima.

Ryan Baker, P. S. (Mayo de 2014). Educational Data Mining and Learning Analytics. 61-76. doi:10.1007/978-1-4614-3305-7_4

S. Aghabozorgi, H. M. (2014). An Approachable Analytical Study on Big Educational Data Mining.

S. Honnutagi, P. (2014). The Hadoop distributed file system. *(IJCSIT) International Journal of Computer Science and Information Technologies, 5*(5), 6238-6243.

S.Honnutagi, P. (2014). The Hadoop distributed file system. *5*(6238-6243).

S.L. González-Ruiz, I. G.-G.-B.-M. (2015). Algoritmos de clasificación y redes neuronales en la observación automatizada de registros. *Servicio de Publicaciones de la Universidad de Murcia, 15*(1).

S.Urmela, M. (Marzo de 2017). Approaches and Techniques of Distributed Data Mining : A Comprehensive Study. *International Journal of Engineering and Technology (IJET), 9*(1). doi: 10.21817/ijet/2017/v9i1/170901408

Sara Landset, T. M. (2015). A survey of open source tools for machine learning with big data in the Hadoop ecosystem.

Shrikant S. Raut, S. S. (2016). A survey on automated deployment of cloudera distribution on Docker containers. *Journal of Information, Knowledge and Reresearch in Computer Engineering, 4*(ISSUE – 01), 823-825. Obtenido de http://www.ejournal.aessangli.in/ASEEJournals/CE168.pdf

Shruti N. Pardeshi, C. P. (2013). Grid Computing Architecture and Benefits. *3*(8).

Skiena, S. S. (2017). *The Data Science Design Manual.* Springer. doi:10.1007/978-3-319-55444-0

Stefan Slater, S. J. (2017). Tools for Educational Data Mining: A Review. *Journal of Educational and Behavioral Statistics, 42*(1), 85–106. doi:10.3102/1076998616666808

Thampi, R. S. (2010). *Survey on Distributed Data Mining.* India. Obtenido de https://arxiv.org/abs/1205.3231

Tsoumakas G, V. (2008). Distributed Data Mining. (2). Obtenido de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.680&rep=rep1&type=pdf

Turnbull, J. (2018). *The Docker Book: Containerization Is the New Virtualization.*

V. Sajwan, V. Y. (Abril de 2015). The Hadoop Distributed File System: Architecture and Internals. *International Journal of Combined Research & Development (IJCRD), 4*(Issue: 3), 541-544.

Verdugo Rodríguez, P. M. (2015). *Análisis de sinergias en despliegue de sistemas de computación distribuida: Apache Spark-OpenStack.* Tesis de maestría, Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingenieros de Telecomunicación, Madrid.

Vuda Sreenivasa Rao, S. V. (2010). Multi Agent-Based Distributed Data Mining: An over view. *(IJCSE) International Journal on Computer Science and Engineering, 2*(4), 1237-1244.

White, T. (2009). *Hadoop: The Definitive Guide.* (M. Loukides, Ed.) O'Reilly.

Yadav, R. (2015). *Spark Cookbook.* Packt Publishing.

Yeo C.S., B. R. (2006). Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers. En *Handbook of Nature-Inspired and Innovative Computing.* Boston: Springer.

Zaki, M. J. (2000). Parallel and Distributed Data Mining: An Introduction. doi:DOI: 10.1007/3-540-46502-2_1

ANEXOS

Anexo 1: Encuesta realizada para medir la satisfacción de los usuarios con el entorno desarrollado

Nombre y Apellidos: _____

Fecha de realización: _____

Perfil en el que se desempeña: _____

Años de experiencia: _____

Categoría docente: _____

Categoría científica: _____

Observación:

Estimado usuario, el objetivo de la presente encuesta está basado en evaluar la propuesta de solución desarrollada con título: “Entorno distribuido para el minado de datos en ambientes educacionales masivos”, donde a través de dicha solución se permitirá contribuir al uso de técnicas de minería de datos distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos, en función de la velocidad y el tiempo de respuesta del procesamiento.

1. ¿Considera usted importante el desarrollo de un entorno distribuido que contribuya al minado de datos en ambientes educacionales masivos?
   Sí_____ No_____ No sé_____
2. ¿Considera que el entorno distribuido desarrollado facilita los trabajos de minería de datos educativos?
   Sí_____ No_____ No sé_____
3. ¿Qué elementos considera positivos dentro del entorno distribuido desarrollado?
   _____
   _____
   _____
4. ¿Qué elementos considera negativo dentro del entorno distribuido desarrollado?
   _____
   _____
   _____
5. ¿Si usted necesitara ejecutar procesos de minería de datos educacionales o facilitar su trabajo en este ámbito de forma eficiente, utilizaría este entorno informático desarrollado?
   Sí_____ No_____ No sé_____
6. ¿Le satisface la composición del entorno para el minado de datos en ambientes educacionales masivos?
   Me satisface mucho_____ Más satisfecho que insatisfecho_____
   Me es indiferente _____ Me insatisface más de lo que me satisface_____
   No me satisface nada_____ No sé qué decir _____

Anexo 2: Cuestionario a Expertos

Nombre y Apellidos: _____

Fecha de realización: _____

Perfil en el que se desempeña: _____

Años de experiencia: _____

Categoría docente: _____

Categoría científica: _____

Observación:

Estimado experto, el objetivo de la presente encuesta está basado en evaluar la propuesta de solución desarrollada con título: “Entorno distribuido para el minado de datos en ambientes educacionales masivos”, donde a través de dicha solución se permitirá contribuir al uso de técnicas de minería de datos distribuida de forma eficiente en modelos de estimación del conocimiento latente aplicados a datos educacionales masivos, en función de la velocidad y el tiempo de respuesta del procesamiento. Por cuanto, sus valoraciones acerca de los asuntos que se someten a su valoración servirán de ayuda para el perfeccionamiento de la propuesta de solución planteada.

El objetivo de la presente encuesta consiste en que usted evalúe cada uno de los indicadores que se le presentarán en la tabla de la subsiguiente sección II, colocando el número en la casilla correspondiente y teniendo en cuenta para ello las siguientes categorías:

5: MUY DE ACUERDO (MA); 4: DE ACUERDO (DA); 3: NI DE ACUERDO NI EN DESACUERDO (Si-No); 2: EN DESACUERDO (ED); 1: COMPLETAMENTE EN DESACUERDO (CD)

II- Lista de indicadores a valorar:

No.	Indicador	Valoración
1.	El entorno distribuido desarrollado tiene utilidad práctica en el minado de datos educacionales masivos.	
2.	Las herramientas, así como la tecnología empleada, brindan novedad a la presente investigación.	
3.	La velocidad de procesamiento de los datos y el tiempo de respuesta se desarrollan dentro del entorno con una correcta eficiencia.	
4.	El entorno distribuido cumple con requisitos necesarios tales como la escalabilidad, la transparencia, la disponibilidad y la tolerancia a fallos.	

5.	La selección y utilización de la infraestructura tecnológica del entorno desarrollado hace posible el análisis de datos educacionales haciendo un uso racional del hardware disponible.	
6.	El entorno distribuido es aplicable para contextos donde sean minados grandes volúmenes de datos educacionales de forma eficiente en modelos de estimación del conocimiento latente.	

III- Si desea exponer cualquier otra opinión, por favor, expréselo en el espacio disponible a continuación:

_____

_____

_____

Anexo 3: Encuesta para determinar el coeficiente de competencia de los expertos seleccionados

Estimado compañero (a): _____

Usted ha sido seleccionado como posible experto para ser consultado respecto a temas relacionados con la minería de datos distribuidos aplicados a datos educacionales masivos , con vistas a la investigación que se está llevando a cabo. Agradecemos sinceramente su valiosa cooperación.

Gracias.

1. Marque con una cruz (X) en la tabla siguiente el valor que se corresponde con el grado de conocimiento que usted posee sobre “minería de datos distribuidos aplicados a datos educacionales masivos”. (Escala ascendente).

0	1	2	3	4	5	6	7	8	9	10

2. Realice una autoevaluación del grado de influencia que cada una de las fuentes que le presentamos a continuación ha tenido en su conocimiento y criterio sobre “minería de datos distribuidos aplicados a datos educacionales masivos”. Marque con una cruz (X) según corresponda en A (alto), M (medio) o B (bajo).

No	**Fuente de argumento**	**Grado de influencia de cada fuente**		
		A(alto)	**M(medio)**	**B(bajo)**
1	Análisis teóricos realizados sobre el tema			
2	Experiencia adquirida en su vida profesional			

3	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales sobre el tema			
4	Conocimiento propio sobre el tema de investigación			
5	Actualización en cursos de postgrado, diplomados, maestrías, doctorado, etc.			
6	Intuición			

Anexo 4: Procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos

El cálculo de dicho coeficiente se realiza de la forma siguiente:

Kcomp = ½ (**Kc** + **Ka**), donde:

Kcomp: Es el Coeficiente de Competencia del experto.

Kc: Es el Coeficiente de Conocimiento o información que tiene el experto acerca del problema, calculado sobre la valoración del propio experto en una escala de 0 a 10, multiplicado por 0.1.

Ka: Es el Coeficiente de Argumentación o fundamentación de los criterios del experto, obtenido como resultado de la suma de los puntos de acuerdo a la siguiente tabla patrón:

Tabla 13: Fuentes de argumentación del conocimiento de los expertos

No.	**Fuentes de argumentación**	**Alto(A)**	**Medio(M)**	**Bajo(B)**
1	Análisis teóricos realizados	0,30	0,20	0,10
2	Experiencia adquirida durante su vida profesional.	0,50	0,37	0,30
3	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales.	0,05	0,04	0,03
4	Conocimiento propio sobre el estado del tema de investigación.	0,05	0,04	0,03
5	Actualización en cursos de postgrado, diplomados, maestrías, doctorado, etc.	0,05	0,04	0,03
6	Intuición.	0,05	0,03	0,02
	Total	1,00	0,70	0,50

Se plantea entonces que:

- ✓ La Competencia del experto es de Alto (A): Si Kcomp > 0,7
- ✓ La Competencia del experto es Medio (M): Si 0,5< Kcomp = < 0,7
- ✓ La Competencia del experto es Bajo (B): Si Kcomp = < 0,5

Anexo 5: Resultado de la encuesta aplicada a los candidatos a expertos para determinar nivel de competencia

Tabla 14: Resultado de la encuesta aplicada a los candidatos a expertos para determinar nivel de competencia. Fuente: Elaboración propia.

Experto	Kc	1	2	3	4	5	6	Ka	Kcomp	CC
1	0,9	0,3	0,37	0,5	0,04	0,03	0,05	0,84	0,87	Alto
2	0,9	0,3	0,5	0,4	0,05	0,04	0,02	0,95	0,925	Alto
3	0,4	0,1	0,3	0,3	0,04	0,04	0,03	0,54	0,47	Bajo
4	0,5	0,1	0,37	0,04	0,03	0,03	0,03	0,6	0,55	Medio
5	0,9	0,1	0,3	0,04	0,05	0,03	0,03	0,55	0,725	Alto
6	0,7	0,2	0,5	0,04	0,04	0,04	0,05	0,87	0,785	Alto
7	0,5	0,1	0,37	0,04	0,05	0,03	0,05	0,64	0,57	Medio
8	0,8	0,2	0,5	0,04	0,04	0,03	0,03	0,84	0,82	Alto
9	0,9	0,2	0,5	0,04	0,05	0,04	0,03	0,86	0,88	Alto
10	0,7	0,2	0,37	0,04	0,03	0,03	0,03	0,7	0,7	Medio
11	0,9	0,3	0,5	0,04	0,04	0,04	0,02	0,94	0,92	Alto
12	0,9	0,1	0,5	0,05	0,05	0,03	0,05	0,78	0,84	Alto

Anexo 6: Respuestas ofrecidas por los expertos para cada indicador

Tabla 15: Respuestas ofrecidas por cada experto en los indicadores determinados.

Experto	Indicador					
1	5	5	4	5	5	5
2	4	5	5	5	5	5
3	4	5	5	3	5	5
4	4	5	5	5	5	5
5	5	3	5	5	5	5
6	4	4	5	4	5	4
7	5	5	4	4	5	5
8	4	3	5	4	5	3
9	5	5	5	3	4	5
10	5	4	5	5	5	4
11	5	5	5	5	5	5