ANÁLISIS COMPARATIVO DEL DESEMPEÑO Y COSTO COMPUTACIONAL DE UNA INFRAESTRUCTURA DE ALMACENAMIENTO Y PROCESAMIENTO DISTRIBUIDO PARA EL PROCESAMIENTO DE COLECCIONES DE TEXTO

YERSON FERNEY PORRAS GARCÍA

UNIVERSIDAD DE LOS LLANOS
FACULTAD DE CIENCIAS BASICAS E INGENIERIA
ESCUELA DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
VILLAVICENCIO, COLOMBIA
2018

ANÁLISIS COMPARATIVO DEL DESEMPEÑO Y COSTO COMPUTACIONAL DE UNA INFRAESTRUCTURA DE ALMACENAMIENTO Y PROCESAMIENTO DISTRIBUIDO PARA EL PROCESAMIENTO DE COLECCIONES DE TEXTO

YERSON FERNEY PORRAS GARCÍA

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

Director:
Ing. ANGEL ALFONSO CRUZ ROA, M.Sc., PhD.

Codirector:
Ing. ROGER CALDERON MORENO, M.Sc.

UNIVERSIDAD DE LOS LLANOS
FACULTAD DE CIENCIAS BASICAS E INGENIERIA
ESCUELA DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
VILLAVICENCIO, COLOMBIA
2018

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

Ing. ANGEL ALFONSO CRUZ ROA, M.Sc., PhD.
Firma del Director

_____

Ing. CÉSAR AUGUSTO DÍAZ CELIS, M.Sc. PhD (c)
Firma del Codirector

_____

Firma de Jurado

"Con todo mi amor y respeto para mi madre, por todo su apoyo incondicional que me ofreció no sólo durante mi estudio universitario sino en toda mi vida y el apoyo que me seguirá ofreciendo hasta el fin de los días.

A mi padre, por todo lo que de alguna u otra forma hizo por mí y me permitió aprender demasiado de la vida y definir quién soy."

AGRADECIMIENTOS

En primera instancia a Dios, por la vida que a diario veo recompensada por experiencias, alegrías, tristezas y oportunidades permitiéndome formar como ser humano, estudiante y ahora, un camino nuevo, profesional.

A mi madre Luz Mery García Martínez, por el apoyo brindado, por aguantarme en mi evolución y desarrollo académico, profesional y como persona, por todas esas noches en que el trabajo ameritaba dejar de lado la comodidad de la cama y ella, con sus preocupaciones y cuidados me acompañaba de alguna u otra forma.

A mi padre Carlos Uldarico Porras Niño, que a pesar de la distancia y los diferentes inconvenientes, aportó a mi vida enseñanzas y vivencias que no cambiaría por nada y que muy seguramente me permitirán ser alguien de bien y un gran profesional.

Al ingeniero Ángel Alfonso Cruz Roa, que me inculcó una gran cantidad saberes que me permitieron llevar a éxito la culminación de mi carrera universitaria y poder convertirme en un excelente profesional y que sin su ayuda y dirección hubiera sido realmente difícil.

Al ingeniero Roger Calderón Moreno, que en mi vida universitaria fue participe en múltiples ocasiones permitiéndome aprender valores éticos y profesionales que me dieron carácter y ser, desde mis primeros semestres.

Al Grupo de Investigación en Tecnologías Abiertas – GITECX y al grupo de estudio Automatic Data-driven Analytics Laboratory – AdaLab, por darme la bienvenida y un grato recibiendo cada día en que llegaba a cumplir con mis labores como estudiante en Proyecto de Investigación cuyo objetivo, luego de tanto, por fin se ve culminado.

A mis docentes e ingenieros que de alguna u otra forma ayudaron en mi formación como profesional y me brindaron apoyo en mi labor, en especial al Ingeniero César Augusto Díaz Celis. A mis compañeros que si bien, ahora somos competencia, me ayudaron a aprender y reforzar mis conocimientos y que, en dicho proceso, se crearon grandes momentos, experiencias y muy buenos recuerdos.

¡Gracias!

TABLA DE CONTENIDO

1 Resumen 8
2 Abstract 9
3 Introducción 11
4 Objetivos 14
4.1 Objetivo general (OG) 14
4.2 Objetivos específicos (OE) 14
5 Marco de referencia 15
5.1 Big Data 15
5.2 Almacenamiento de Datos 16
5.2.1 Bases de Datos No SQL 17
5.3 Procesamiento Distribuido 18
5.4 Apache Hadoop 18
5.4.1 Apache HBase 19
5.4.2 Apache Hadoop Distributed File System – HDFS 20
5.4.3 Algoritmo MapReduce 21
5.5 Apache Spark 22
5.6 Procesamiento de documentos de Texto 25
5.6.1 Bolsa de Palabras - *Bag of Words* (BoW) 25
5.6.2 Búsqueda de documentos por Relevancia 29
5.7 Feature Driven Development – FDD 29
5.8 Desempeño Computacional 30
5.8.1 Speed Up 30
5.8.2 Ley de Amdahl 31
6 Desarrollo de la metodología 31
6.1 Metodología de Investigación 31
6.2 Metodología de Desarrollo de Software 32
6.2.1 Modelo General del proyecto 32
7 Aplicación Búsqueda de Documentos por Relevancia 35
7.1 Módulos de la Bolsa de Palabras 36
7.1.1 Etapa 1: Creación de Bag of Words 36

7.1.2 Etapa 2: Procesamiento y Comparación de Documentos 36
7.1.3 Etapa 3: Búsqueda y Relevancia de Documentos. 36
7.2 Implementación BoW en Apache Spark.....37
7.2.1 Core (scriptCore.py).....37
7.2.2 Corpus (scriptCorpus.py) 37
7.2.3 BoWGral (scriptBoWGral.py) 37
7.2.4 DiccionarioGral (scriptDiccionarioGral.py).....37
7.2.5 BoWIndividual (scriptBoWIndividual.py).....38
7.2.6 Histogramas (scriptHistogramas.py) 38
7.2.7 idDoc (scriptIdDoc.py).....38
7.2.8 idWord (scriptIdWord.py) 38
7.2.9 SparseMatrix (scriptSparseMatrix.py) 38
7.2.10 TFIDFMatrix (scriptTFIDFMatrix.py).....38
7.2.11 Producto Punto (scriptProductoPunto.py) 39
8 Resultados.....39
8.1 Conjunto de Datos 39
8.2 Diseño Experimental.....40
8.3 Resultados preliminares.....42
8.4 Resultados finales.....46
8.5 Análisis de Resultados.....55
8.5.1 Desviación Estándar 55
8.5.2 *SpeedUp* Descendente.....56
8.5.3 *SpeedUp* Ascendente 56
9 Conclusiones 58
10 Recomendaciones.....58
11 Anexos y Repositorios.....59
11.1 Plataforma Proyecto: 59
11.2 Plataforma Apache Spark: 59
11.3 Plataforma Apache Hadoop: 60
12 Referencias bibliográficas 60

1 Resumen

La computación distribuida para el procesamiento en paralelo sobre un clúster de computadores ofrece un rendimiento computacional mayor con tiempos de respuesta mucho menores que si se realizara el mismo procesamiento en una sola máquina. Si se toma en cuenta que cada computador por sí solo implica una serie de recursos y costos para su funcionamiento (económicos, humanos, etc.), y además se considera que cuanto más computadores se incluyan al clúster representará un aumento en tiempos de transmisión de información entre ellos, no es difícil inferir que habrá un punto en el cual se deberá hacer un balance entre las ventajas que ofrecen la inclusión de más computadores (en términos de capacidad de cómputo) y el costo que representan (en energía, dinero y transferencia de datos).

Por lo tanto, el presente trabajo tuvo como objetivo realizar un análisis comparativo de la implementación de una plataforma de procesamiento distribuido de una colección de documentos de texto para la construcción de un sistema para la búsqueda de información y relevancia de resultados, usando representaciones como *Bag of Words (BoW)* y producto punto por medio de algoritmos de procesamiento distribuido, los cuales permitieron identificar la relación entre mejora de la capacidad de cómputo cuando se incrementa el número de computadores en el clúster como apoyo a la toma de decisiones en unidades de computación de alto desempeño para el análisis masivo de datos.

El trabajo se realizó por medio de plataformas dedicadas tanto para el almacenamiento como el procesamiento distribuido, implementando para ello Apache Hadoop, más exactamente *Hadoop Distributed File System (*HDFS); y Apache Spark, respectivamente. Estructuralmente el clúster contó con 17 equipos conectados y vinculados entre sí, en los cuales se almacenó un conjunto de datos (dataset) de 500 archivos de texto plano de tamaños variables con HDFS, mientras que con Apache Spark se implementó una serie de algoritmos de las fases de BoW para resolver las tareas programadas (e. g. Creación de un Core de documentos de solo palabras, de un

Diccionario general de palabras referentes al *dataset*, de histogramas, de matrices con valores relacionados con los documentos y sus palabras existentes, cálculo de ecuaciones matemáticas para la búsqueda de palaras, entre otros) las cuales fueron evaluadas variando el número de equipos que procesaron el conjunto de datos, demostrando comportamientos positivos y negativos en diferentes escenarios propuestos y fases de BoW en el entorno distribuido.

La obtención de resultados con diferentes comportamientos según el tipo de algoritmo y la carga computacional, sugiere la ventaja de la computación distribuida para el procesamiento masivo de datos (*Big Data*), así como sus puntos para mejorar brindando una oportunidad de indagar en posibles causas, mejoras y desarrollos futuros a trabajos y escenarios afines con el fin de adquirir capacidades de análisis de datos (*Data Analytics*) como apoyo a las necesidades de investigación, educativas y productivas en la región.

Palabras clave: *Procesamiento Distribuido, Apache Spark, Bolsa de Palabras, Clúster, Balance costo-beneficio, Búsqueda de Información*.

2 Abstract

The distributed computing for parallel processing on a computer cluster offers greater computational performance with less response times in comparison to the secuencial processing. If it is taken into account that each computer alone by itself implies a series of resources and costs for its operation (economic, human, etc.), and it is also considered that the more computers are included to the cluster, it will represent an increase in information transmission times among them, it is not hard to infer that there will be a point at which a balance should be made between the advantages offered by the inclusion of more computers (in terms of computing capacity) and the cost that they represent (energy, money and data transfer).

Therefore, the present work aimed to perform a comparative analysis of the implementation of a distributed processing platform of a collection of text documents for the construction of a system to search for information and result relevance, using representations like Bag of Words (BoW) and dot product by means of distributed processing algorithms, which allowed to identify the relation between improvement of the computing capacity when the number of computers in the cluster is increased as support to the decision making in high performance computing units for the massive analysis of data.

The work was done through dedicated platforms for both, storage and distributed processing, implementing Apache Hadoop, more exactly Hadoop Distributed File System (HDFS); and Apache Spark, respectively. Structurally, the cluster had 17 computers connected and linked to each other, in which a dataset of 500 flat text files from variable sizes were stored with HDFS, while a series of algorithms for the BoW phases were implemented with Apache Shark to solve the scheduled tasks (e. g. Creation of a Core of single-word documents,of a general Dictionary of words referring to the dataset, of histograms, of matrices with values related to the documents and their existing words, calculation of mathematical equations for the search of words, among others) which were evaluated varying the number of computers that processed the dataset, demonstrating positive and negative behaviors in different given scenarios and BoW phases in the distributed environment.

The result obtention with different behaviors according to the type of algorithm and the computational load, suggests the advantage of the distributed computation for the massive data processing (Big Data), as well as its point to improve providing an opportunity to investigate in possible causes, improvements and future developments to works and related scenarios in order to acquire data analysis capabilities (Data Analytics) as support to the educational and productive in the region needs of the research.

***Key words**: Distributed processing, Apache Spark, Bag of Words, Cluster, Trade-off, Information Retrieval.*

3 Introducción

En la actualidad un clúster de servidores o computadores puede realizar procesamiento en paralelo para la distribución de carga computacional de tareas que son independientes entre sí permitiendo mejorar el rendimiento y los tiempos de procesamiento al ejecutar un mayor número de instrucciones por unidad de tiempo. Por lo tanto, en principio se debería dar que, a mayor número de servidores para procesamiento distribuido en paralelo, menor deberá ser el tiempo de procesamiento[1]–[5].

Por otro lado, la implementación de una infraestructura de procesamiento distribuido implica que se debe contar con software y algoritmos eficientes encargados de la lógica y operaciones asociados con la distribución de la carga computacional, así como de un conjunto de recursos de hardware instalados y configurados en dicho clúster, tanto para el almacenamiento como el procesamiento de la información [2], [4], [5].

Disponer de un equipo de cómputo, e.g. un servidor, significa que éste requiere de energía eléctrica para su funcionamiento, un adecuado y frecuente mantenimiento del equipo, además, de la compra de diferentes componentes de software y hardware que permiten su correcta configuración e integración al clúster. Por ejemplo, licencias de software privativo, cableado, dispositivos de red, UPS's, entre otros [4], [6], [7]. Tales necesidades repercuten en recursos adicionales asociados a los costos de la infraestructura del clúster, Por lo tanto, requerir de más servidores, aumentará igualmente el costo de ejecución y producción desde un punto de vista económico [8].

Debido a que el aumento en rendimiento computacional y la reducción en tiempo de procesamiento de un clúster no son lineales en relación al número de servidores

instalados [9], puesto que, a mayor número de nodos, mayor es el tiempo que se requiere en la transmisión de datos entre ellos, la ganancia en el desempeño computacional del procesamiento distribuido tiende a comportarse de forma logarítmica [9].

Por otro lado, El *Big Data* se caracteriza por la masiva cantidad de datos que se crean y almacenan a diario a niveles de petabytes [10], surge la necesidad de buscar una forma en que los datos puedan ser almacenados en más de un disco o equipo, por lo que se crea el sistema de archivos distribuido [11]. Ahora, dichos datos requieren ser procesados e interpretados para obtener información relevante, por lo que se hace indispensable tener un sistema de procesamiento distribuido [12]. Un claro ejemplo de esto, serían las aplicaciones de búsqueda y análisis de texto. Dichas implementaciones demandan un procesamiento en cada uno de los datos almacenados por lo que, si se contara con un solo servidor, se tomaría mucho más tiempo a que se hiciera el mismo procesamiento en varios servidores por lo que se aprovecharían las Unidades de Procesamiento - CPU de cada uno de ellos para procesar parte del conjunto de datos, dar resultados preliminares y finalmente dar un único resultado [13].

Cabe mencionar, que muchas Instituciones de Educación Superior - IES, e incluso de otra índole, cuentan con salas de cómputo dotadas de varios computadores conectados en red que podrían ser usados como un clúster de computadores heterogéneo y así, ofrecer un sistema de procesamiento distribuido para tareas que requiera la institución, e. g. un sistema de procesamiento de texto que detecte plagio de documentos, minería de datos, procesos experimentales relacionados con investigación científica, entre otros [14], [15].

Se presenta un proceso detallado que argumenta criterios de selección de rendimiento y procesamiento de plataformas de software que, de manera clara, permiten medir y calificar cuantitativamente procesos de ejecución [3], [12], [16]. En consecuencia, este trabajo sirve como punto de referencia para analizar la capacidad de un clúster de computadores y saber hasta qué punto puede costar la inclusión de cada nodo para el

procesamiento distribuido con el fin de optimizar el uso de recursos, energía eléctrica, red y comunicación, espacio físico, mantenibilidad, entre otros, buscando un mayor beneficio en tiempos de procesamiento computacional, rendimiento de operación y transmisión, y una reducción en costos de producción y ejecución, es decir, un aumento en las ganancias económicas o reducción de costos a las que diera lugar [7], [17]–[20].

Este trabajo pretende servir como guía para la realización de otros trabajos al ofrecer una metodología de investigación y de desarrollo de software ajustables, incluso, a otras áreas u objetivos diferentes, como ciencia, ingeniería, literatura, educación, medicina, entre otras [21]–[23]. Por lo que, se abren las puertas a muchos trabajos relacionados con *Big Data* y procesamiento de datos tanto distribuido como en paralelo que quieran empezar teniendo como objetivo el máximo aprovechamiento del sistema con una inversión económica pertinente. Trabajos que pueden ser aplicados, e. g. en la educación, ya que este trabajo ofrece una plataforma con funcionalidades para el procesamiento, la comparación y la similitud de documentos, las cuales son piedra angular para aplicaciones cuya ejecución optimiza recursos y costos al ejecutarse en un clúster de servidores e. g., y no se limitan a, un sistema detector de plagio, un sistema de recomendación de contenidos textuales según búsquedas hechas, un sistema para el análisis y representación del comportamiento de documentos [9], [10], [15], [24]–[27].

Para el desarrollo y culminación de este trabajo se planteó iniciar con un proceso experimental con 17 nodos de los cuales 16 eran nodos encargados de ejecutar tareas o procesos, llamados nodos esclavos; y 1 era el encargado de coordinarlos y asignar la distribución de carga, nodo Maestro. Cada proceso requería de la ejecución de una aplicación de búsqueda por medio de varios scripts que generaban resultados siendo almacenados directamente en el HDFS para su posterior uso, de tal forma que la carga de toda la aplicación fuera fraccionada y reducida a partes procurando optimizar recursos y previniendo fallos en los sistemas (e. g. fallos eléctricos o apagones, sobrecarga de memoria, exceso de flujo de datos en red, etc).

Al ejecutar los scripts en todos escenarios experimentales con 1, 2, 4, 8 y 16 nodos esclavos, se obtuvieron resultados cuyos comportamientos demostraron circunstancias que permitieron deducir una serie de hipótesis e ideas relacionadas con la optimización y manejo adecuado de los recursos físicos, de software y de programación como tal. Se vieron aumentos e incluso, disminuciones en los *SpeedUp* tomados indicando que, en lugar de disminuir el tiempo de procesamiento, aumentaba. A lo que se dispuso a analizar causas y respuestas planteadas en detalle al final de este trabajo.

4 Objetivos

4.1 Objetivo general (OG)

- Implementar en una plataforma de procesamiento y almacenamiento distribuido el procesamiento de colecciones de documentos de texto para la búsqueda de información usando la metodología de representación de documentos conocida como *Bag of Words* usando el algoritmo *MapReduce*.

4.2 Objetivos específicos (OE)

- Seleccionar e implementar una plataforma de procesamiento y almacenamiento distribuido con múltiples nodos para el procesamiento distribuido del algoritmo *MapReduce*.
- Implementar la metodología de *Bag of Words* para el procesamiento y representación de colecciones de documentos de texto usando el algoritmo *MapReduce*.

- Implementar un sistema de búsqueda de información por relevancia usando la representación *Vector Space Model* de los documentos y la medida de similitud Coseno a partir de la representación de *Bag of Words*.
- Evaluar los resultados referentes a rendimiento computacional, tiempos de respuesta, costos de producción y ejecución en diferentes escenarios y número de nodos vinculados al clúster.

5 Marco de referencia

5.1 Big Data

Big Data se define como un conjunto de técnicas, tecnologías, herramientas y metodologías necesarias para el procesamiento de una masiva cantidad de datos con el objetivo de conseguir información analítica y representativa de todos los datos en su conjunto. Adicionalmente, *Big Data* presenta una serie de características como se puede observar en la figura 1 [10], [27]–[29].

- ➔ **Volumen**: Se refiere a la cantidad desmesurada de datos que se crean, incluso a diario. Un claro ejemplo de esto son todas las aplicaciones Web como redes sociales que cuentan con un registro de varios petabytes diariamente de almacenamiento [28].
- ➔ **Variedad**: Al crearse a diario muchísimos datos, también se considera la forma en que se representan ya sea si son estructurados o no, como por ejemplo los blogs, archivos de búsquedas, redes sociales, búsquedas y demás [28].
- ➔ **Veracidad**: Tomando la definición de [18]: *"la Veracidad hace referencia a los datos previamente evaluados y filtrados de datos irrelevantes o "basura" a los que se les hace minería y análisis. Se relaciona con otras características*

especiales como Validez y Volatilidad que corresponden a la exactitud de los datos relevantes y al tiempo en que se consideran válidos los datos, respectivamente".

- **Valor**: Representa el valor, peso, relevancia o importancia de la información que puede obtenerse al analizar e interpretar los datos para una empresa. Para muchos analistas, Valor comprende la unión de Veracidad, Validez y Volatilidad [18].
- **Velocidad:** Corresponde a la velocidad con que se crean datos, cuya necesidad de procesamiento es alta y se requiere que sus tiempos de respuesta, ante diferentes tipos de peticiones, sean mínimos. Se puede ver reflejado en empresas con servicios en la web como redes sociales, comercio electrónico y demás [30].

Figura 1. Características que definen los datos masivos (*Big Data*). Tomada de [10].

5.2 Almacenamiento de Datos

Hoy en día, el almacenamiento de datos se puede dar en lo que se denomina bases de datos, la cuales están clasificadas en dos tipos: Bases de Datos Relacionales (*Relational Data Bases* – RDB o RDBMS) y las Bases de Datos No SQL (*Not Only SQL*

Data Bases) [11], [28]. El almacenamiento realizado en cualquiera de los dos tipos de bases de datos afecta directamente en el rendimiento y tiempos de respuesta, por lo que se ha comprobado que para almacenar grandes volúmenes de datos y procesarlos, las Bases de datos No SQL son más eficientes que las RDB [11].

5.2.1 Bases de Datos No SQL

Existe una clasificación para los tipos de bases de datos No SQL que surge a partir de la estructura de almacenamiento interno. De manera generalizada mencionaremos los tipos y algunos ejemplos de cada una:

- ➔ Tabular, Columnar o familia de columnas: DynamoDB, BigTable, HBase.
- ➔ Orientada a documentos: CouchDB, MongoDB, BaseX.
- ➔ Clave-Valor (*key-value*): Cassandra, Voldemort, Riak.
- ➔ Grafos: Neo4j, AllegroGraph.

Algunas bases de datos No SQL pueden presentar estructuras mixtas por lo que pueden encontrarse clasificadas, incluso en más de un tipo [11], [28]. Por ejemplo, algunos autores clasifican a Cassandra como tipo Clave-Valor y otros, como familia de columnas. Caso similar sucede con HBase [9], [28], [31].

Las No SQL permiten un escalado horizontal de fácil integración, tiempos de respuesta bajos y un aumento en rendimiento en el procesamiento y almacenamiento de grandes cantidades de datos, *Big Data* [11], [19], [28], [29], [32]**,** en consecuencia, las bases de datos No SQL del tipo Clave-Valor se usaron de forma transparente en este trabajo, más exactamente Apache HBase a causa de que Hadoop Distributed File System (HDFS) implementa HBase de forma autónoma para almacenar los datos como archivos distribuidos [33].

5.3 Procesamiento Distribuido

El presente proyecto se centró en un procesamiento distribuido y no en paralelo [14], [34], es decir, se buscó implementar un clúster de computadores en donde todos los nodos trabajen y procesen datos cuando se requieran y en cambio, no es un obligación o restricción que todos lo hagan al mismo tiempo.

5.4 Apache Hadoop

Este *framework* ofrece una serie de herramientas que permiten no solo el almacenamiento masivo de datos, sino también su procesamiento, análisis, facilidad de codificación y demás. Hadoop es frecuentemente usado en áreas de *Big Data* como *Business Intelligence, Data Science, Data Mining* y *Machine Learning* (*e.g*. Apache Mahout) [9], [29]. Está implementado en Java y hace uso del *Java Runtime Environment* 1.6 (JRE 1.6), o superior, y de *Secure SHell* (SSH) [4].

Apache Hadoop se divide en dos capas, ver figura 2:

- Apache *MapReduce*: Se basa en el algoritmo desarrollado por Google. Se encarga de todo lo referente a ejecución de tareas de la aplicación [4]. Se explicará nuevamente en una sección propia para el algoritmo *MapReduce.*
- Apache HDFS: Se encarga de la gestión y del almacenamiento distribuido de los datos en el clúster [4].

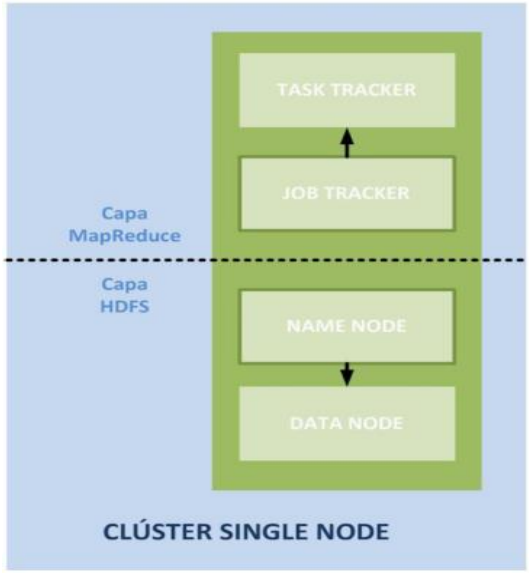


Figura 2. Estructura de un Clúster Hadoop para un solo nodo. Tomada de [4].

5.4.1 Apache HBase

Según [28], es una base de datos de código abierto (*OpenSource*) cuyo almacenamiento se da por clave valor (*key-value*) o como familia de columnas, siguiendo el modelo *Big Table* de Google [19], [28], [29], [31]. Hace parte de Hadoop, una plataforma de la Apache Software Foundation [9]. En [9], se mencionan algunas características importantes por las cuales HBase se implementó en este proyecto. En resumen:

- Almacenamiento tolerante a fallos.
- Apoyo a tareas MapReduce.
- Alta disponibilidad.
- Replicación a través de HDFS.
- Fragmentación y balanceo de carga.
- Búsquedas cercanas a tiempo real.

5.4.2 Apache Hadoop Distributed File System – HDFS

Es un sistema de archivos distribuidos inspirado en *Google File System* (GFS) y *MapReduce* de Google [35]. Hace parte de la plataforma Hadoop desarrollado por Doug Cutting de la Apache Software Foundation, cuyas características mencionadas en [4] destacan:

- ✓ Soporta aplicaciones que procesan sobre grandes cantidades de datos de manera distribuida.
- ✓ Diseñado para aplicaciones que trabajen con amplia cantidad de nodos.
- ✓ Alta flexibilidad y escalabilidad permitiendo implementar clústeres con diferentes topologías.

En [28] definen de forma concreta la estructura y funcionalidad del HDFS a lo que se citan a manera de lista:

- *"HDFS está diseñado para sistemas que realicen pocas escrituras y en cambio, requieren de bastantes lecturas de datos."*
- *"Permiten tolerancia a fallos debido a su sistema de replicación de datos."*
- *"La arquitectura de HDFS se compone de dos tipos de nodos, un nodo principal, o maestro, llamado NameNode y de varios nodos esclavos, los DataNodes".*
- *"El NameNode se encarga de coordinar y almacenar los metadatos del clúster, como nombres, direcciones, distribuciones de los datos almacenados en los diferentes DataNodes y controlar el acceso de los diferentes archivos de usuarios. El NameNode se encarga de abrir, cerrar, mover, nombrar y renombrar archivos y directorios".*
- *"Los DataNodes, gestionan la lectura y escritura de los datos de los usuarios, además de realizar la replicación según indicación del NameNode. En un clúster pueden existir miles de clientes HDFS y miles de DataNode, debido a que cada DataNode puede ejecutar múltiples tareas de aplicaciones de forma simultánea".*

En la figura 3 se puede observar un claro ejemplo de la estructura topológica que constituye HDFS con algunas de sus relaciones y funcionalidades.

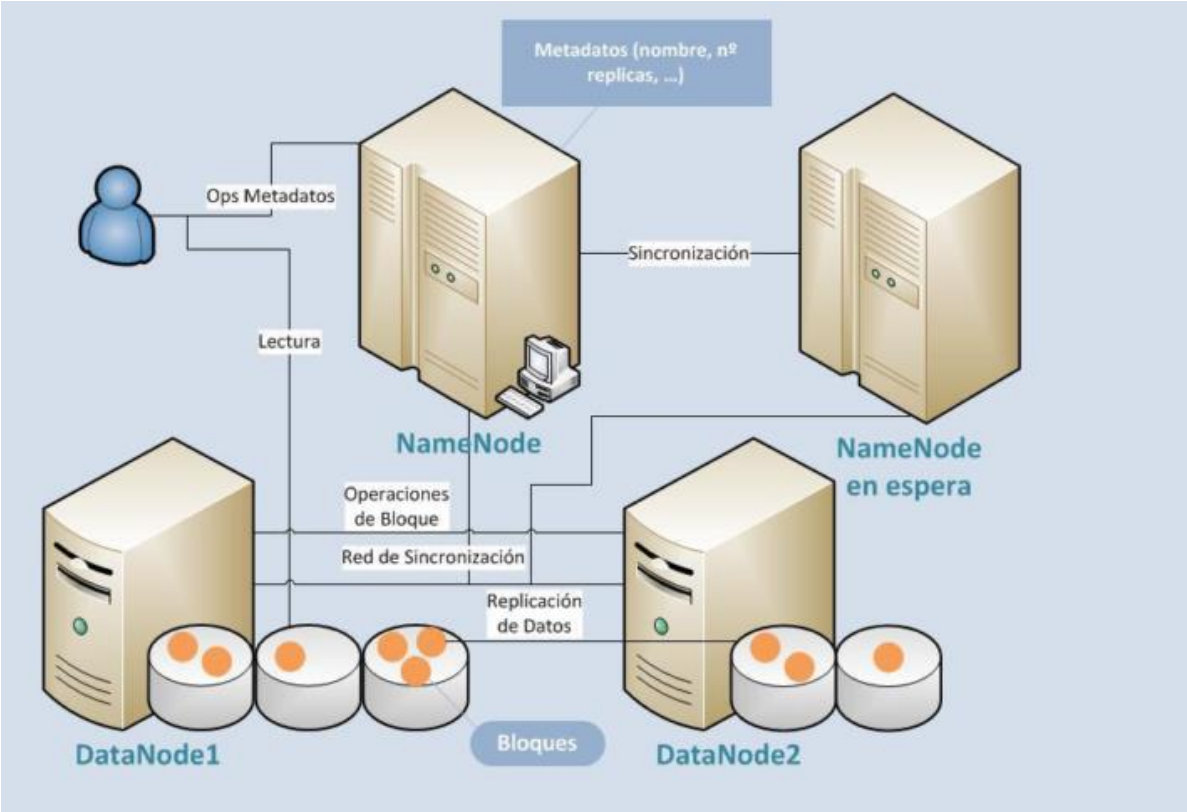


Figura 3. Topología HDFS. Tomada de [4].

5.4.3 Algoritmo MapReduce

Es un algoritmo diseñado por Google para el procesamiento paralelo de grandes conjuntos de datos, posteriormente fue implementado por Doug Cutting para el desarrollo de Apache Hadoop y sus complementos, siendo para este caso Apache MapReduce [29]. En un principio, MapReduce se creó para procesar una gran cantidad de datos en un solo equipo [28]. *"Su estructura compone un maestro, llamado JobTracker y un esclavo, TaskTracker, por cada nodo. El JobTracker se encarga de programar las tareas, los componentes que manejan el esclavo, y éste ejecuta las tareas según indicaciones del JobTracker"* [28]. MapReduce divide su funcionamiento en 2 tipos de operaciones, la operación de mapear, llamada función *map()* y la función

de reducir, *reduce()*. La primera se encarga de procesar todos los datos y representarlos en clave-valor cuyo resultado será recibido por el *reduce()* para procesarlos una última vez y así dar una respuesta definitiva. Cada función se puede programar en lenguajes como Java, Python, C++ y Ruby [28]. Se puede observar una representación del flujo de trabajo en la figura 4.

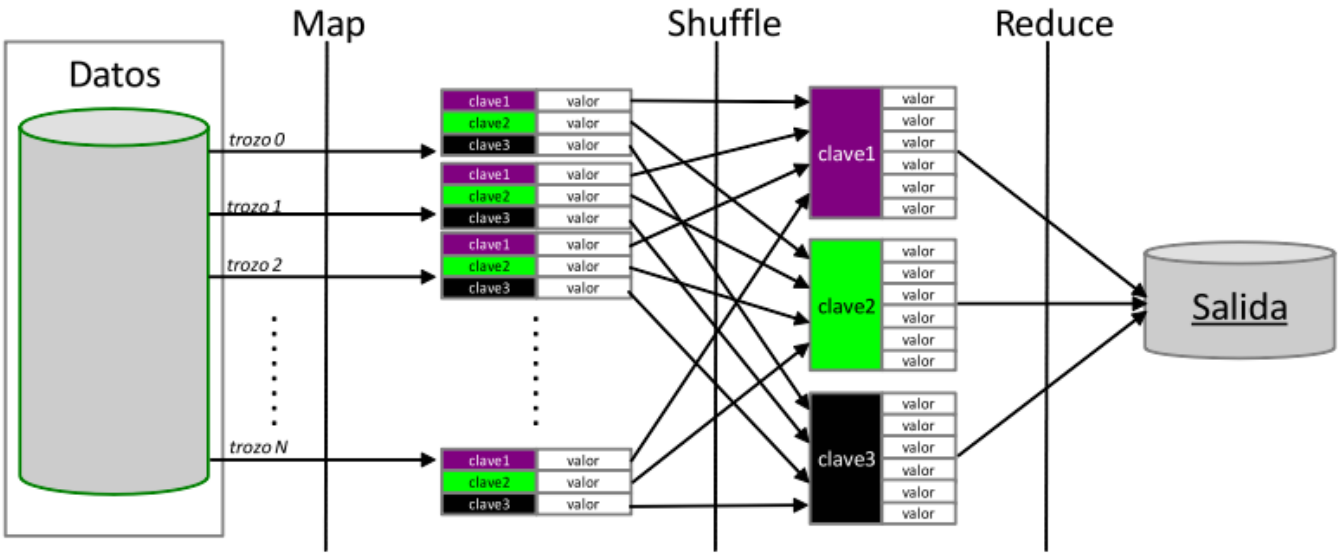


Figura 4. Flujo de trabajo de *MapReduce.* Tomada de [36].

5.5 Apache Spark

Es un *framework* que permite el procesamiento masivo de datos en clústeres, el cual proporciona una Interfaz de Programación de Aplicaciones (*Application Programming Interface* - API) de alto nivel en Java, Scala, y Python [20], [37]. Compatible con Hadoop [38] y con demostraciones de su superioridad en rendimiento y tiempos de respuesta, ya que soluciona algunas de sus limitaciones al permitir ejecución y procesamiento en memoria [3], [10], [20], [24], ver figura 5. Apache Spark permite igualmente la implementación del algoritmo *MapReduce*.

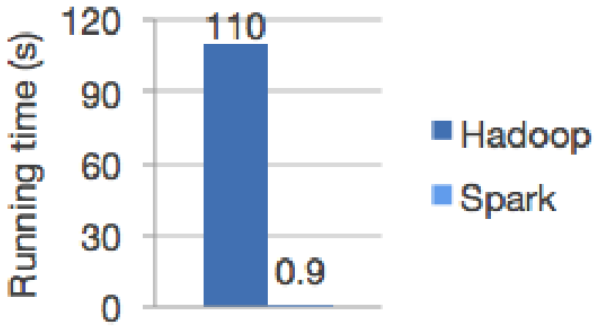


Figura 5. Tiempos de ejecución Hadoop vs Spark. Tomada de [15].

Apache Spark se compone de tres partes [16]:

- ✓ Spark: Proporciona una API para realizar aplicaciones de alto nivel. Spark se encarga del procesamiento paralelo distribuido de datos en un clúster.
- ✓ Clúster: Spark hace uso de un clúster que debe ofrecer todos los recursos necesarios para que pueda funcionar.
- ✓ Almacenamiento: Implementa HDFS como su sistema de archivos distribuido.

Spark ofrece un catálogo de herramientas de alto nivel que facilitan el desarrollo de aplicaciones escalables. Se puede encontrar una definición más detallada de cada una en [3]. Ver figura 6.

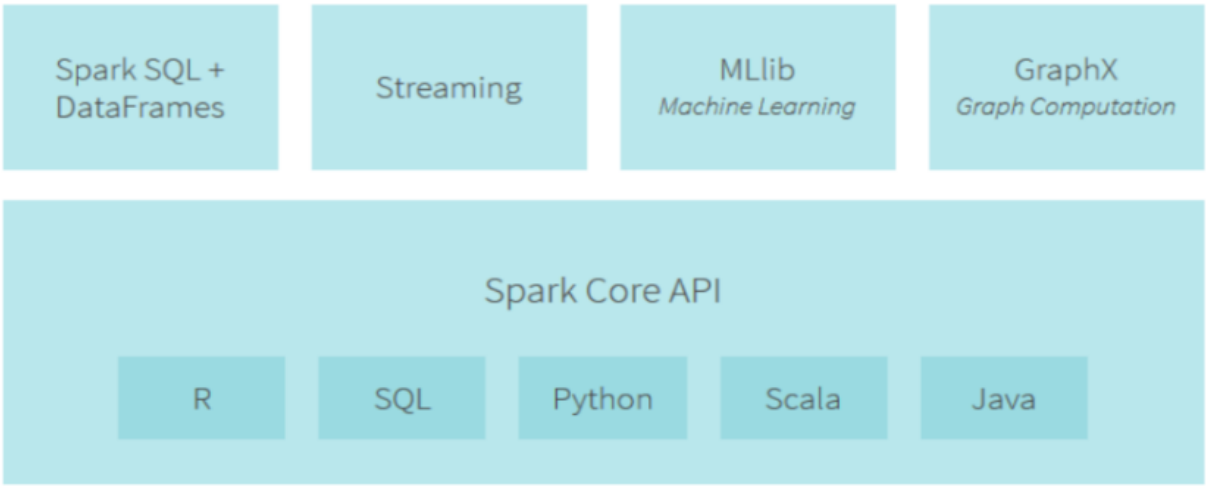


Figura 6. Esquema del núcleo de Apache Spark y sus librerías. Tomada de [3].

Spark, estructuralmente se divide en nodos esclavos (*Worker Nodes*), un nodo maestro (*Driver Program*) y un administrador del clúster (*Cluster Manager*), en [3] se detalla bien el proceso de comunicación de Spark, en la figura 7 se resume este proceso.

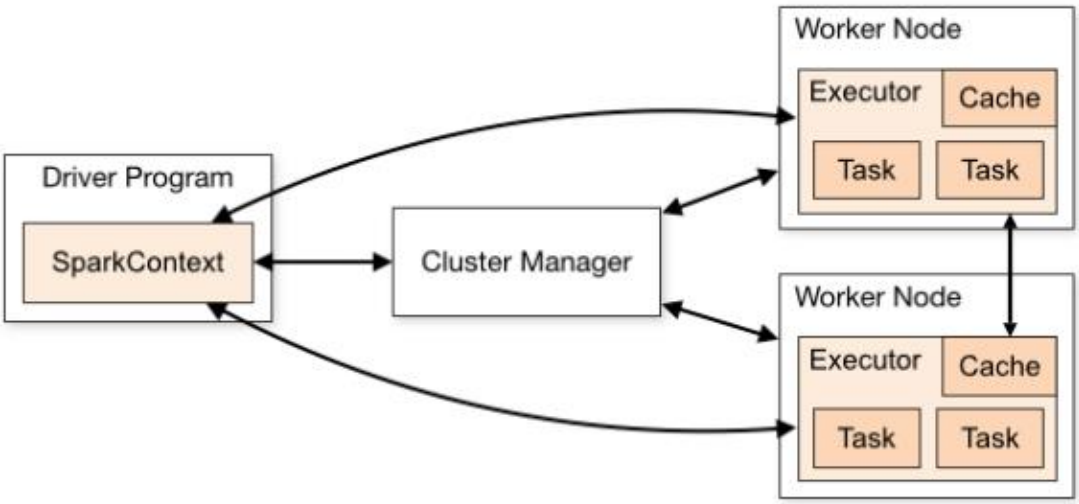


Figura 7. Esquema que muestra la comunicación entre los elementos de Spark dentro de un clúster. Tomada de [3].

5.6 Procesamiento de documentos de Texto

5.6.1 Bolsa de Palabras - *Bag of Words* (BoW)

Es una colección de palabras representadas en un vector por cada documento, o una matriz para un grupo de documentos, en donde cada valor del vector representa una palabra, ya sea como frecuencia, peso o existencia en el documento, procesos que se conocen como *Word frequency weighting, tf*idf weighting* y *Boolean weighting*, respectivamente [39]. Existen dos técnicas de representación usadas por BoW: La primera se denomina *stemmer* (lematizadores) o *stemming* y consiste en tomar cada palabra y representarla en su raíz. La segunda, *Stopwords*, en donde se filtran las palabras relevantes eliminando aquellas que no tengan un valor semántico como tal [40]. Se elaboró un diagrama de la representación BoW, véase la figura 8.

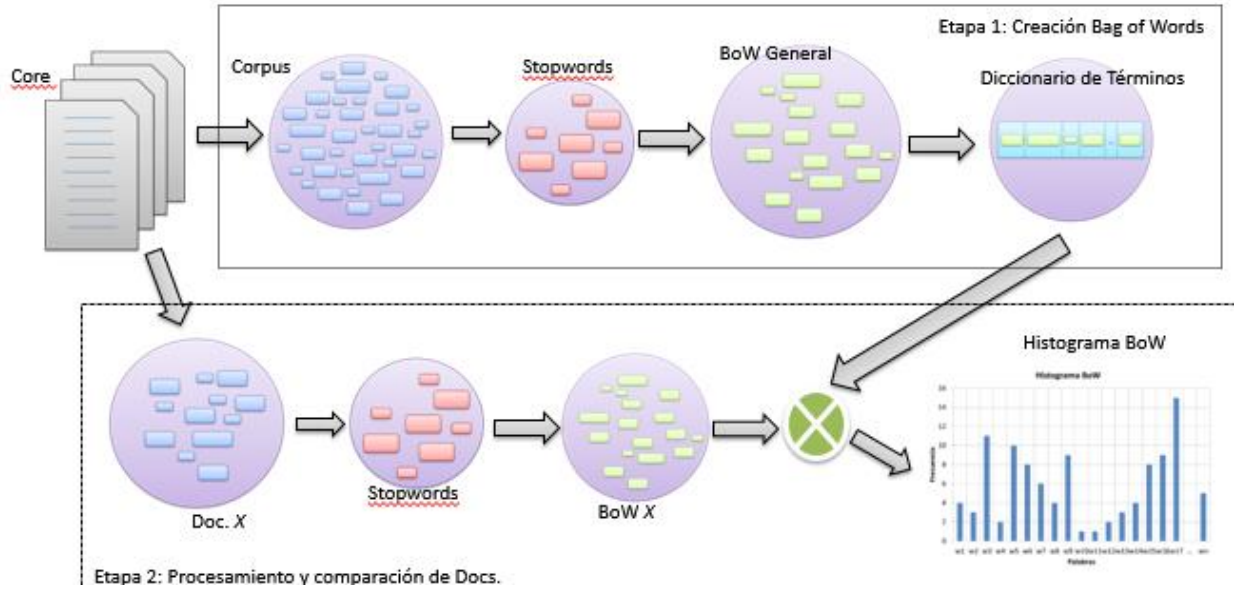


Figura 8. Diagrama general de la implementación de la representación de documentos, *Bag of Words*. Fuente propia.

5.6.1.1 Stop Words

Es un conjunto de palabras cuyo significado semántico no representa información alguna y son usados para conectar o preceder otras palabras. Por lo general, son preposiciones, artículos, conectores, disyunciones, conjunciones, y demás. Se usa como filtro para eliminar palabras irrelevantes de la representación BoW [40].

5.6.1.2 Histograma *Bag of Words* (BoW)

Muestra la relación de las frecuencias de las palabras de la representación BoW del documento a procesar y las palabras del BoW general perteneciente a todo el *corpus* de documentos almacenado [41]–[43]. Véase un ejemplo en la figura 9.

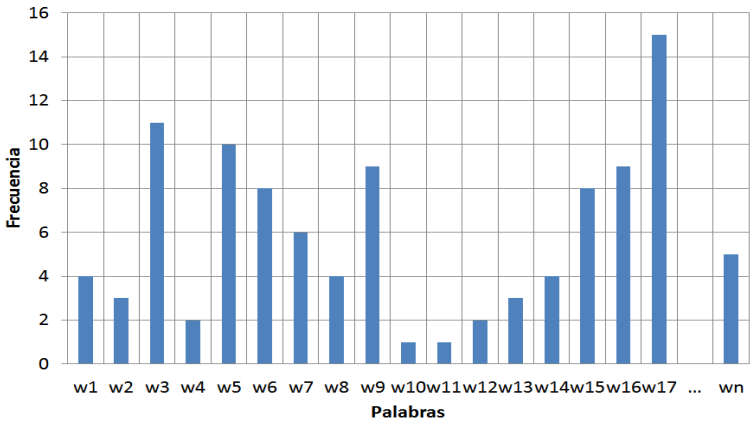


Figura 9. Histograma BoW. Fuente propia.

5.6.1.3 Diccionario de Términos

Un diccionario de términos es una lista de las palabras únicas obtenidas a partir de la colección de datos cuya presencia del término significa la existencia en el *corpus* de los datos procesados.

5.6.1.4 Matriz Término/Documento

"Es una matriz de M-términos por N-documentos. Cada documento se representa como un vector fila d en el espacio de términos tal que d = { w_1, w_2, …,w_m }, donde w_{ij} es igual a la frecuencia del término (conocido como TF) normalizado en la colección multiplicado por la inversa de la frecuencia del documento (conocido como IDF) para ese término, en lo que se conoce como el valor TF-IDF" [44].

$$W_{i,j} = \frac{frecuencia_{i,j}}{\max(frecuencia_i)} * \log\left(\frac{N}{1+n_j}\right) \quad (1)$$

Para este trabajo se ajustó el valor TF-IDF de tal forma que en ningún caso se fuera a obtener una indeterminación al dividirse por cero. Se agregó un valor que fuera directamente relacional al N para que afectara en lo más mínimo la ecuación. La ecuación final usada fue la siguiente:

$$W_{t,q} = tf_{t,q} * \log\left(\frac{D+\log D}{d_t}\right) \quad (2)$$

Donde D es el número de documentos, $tf_{t,q}$ es la frecuencia del término en la consulta y d_t es el número de documentos que tengan ese término. Véase un ejemplo de la estructura general en la figura 10.

	t_1	t_2	…	t_j	…	t_F
d_1						
d_2						
…						
d_i				$W_{i,j}$		
…						
d_N						

Figura 10. Matriz de Términos por Documentos. Tomada de [44].

Existe otra representación más simple en donde cada valor de la tabla representa únicamente la existencia de cada palabra en cada documento, dicho valor es binario, siendo 1 para existencia y 0 para ausencia, la cual se representa con una matriz dispersa término-documento. Véase un ejemplo en la figura 11.

F/D	d1	d2	d3	d4	d5	...	dj
w1	1	0	1	0	1	...	1
w2	0	0	1	0	1	...	1
w3	0	1	0	1	1	...	0
w4	0	0	0	1	1	...	1
w5	0	0	1	0	0	...	1
...	...	...	...	...	...	...	...
wn	0	0	1	1	1	...	0

Figura 11. Matriz Dispersa Término-documento. Fuente propia.

5.6.1.5 Índice Invertido

Consiste en un matriz de dos columnas y n-filas por lo que se puede observar como una *Hash-Table*, una lista de listas o un arreglo de listas, su estructura es similar a los clave-valor, donde cada fila representa una palabra y los documentos en lo que se encuentra, en otras palabras, la clave sería la palabra y el valor sería una lista de documentos en los que aparece [46]. Véase un ejemplo en la figura 12.

w1	d1.
w2	d3, d8, d10, d11, d19.
w3	d2, d3.
w4	d5, d8.
w5	.
w6	d7, d8, d16, d45, d58, d70.
...	...
wi	d2, d14.

Figura 12. Índice Invertido. Fuente propia.

5.6.2 Búsqueda de documentos por Relevancia

Una aplicación de la *Bag of Words* es un sistema de búsqueda de documentos por relevancia en donde a partir de valores numéricos se priorizan resultados descendentemente. Una representación general se puede visualizar en la figura 13.

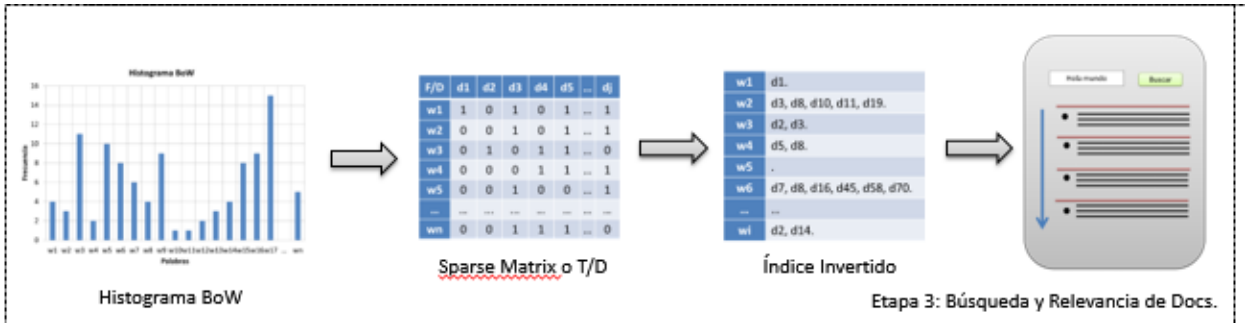


Figura 13. Representación general de una aplicación de búsqueda y relevancia de palabras y documentos. Fuente propia.

5.7 Feature Driven Development – FDD

Es una metodología de desarrollo ágil de software centrada en rasgos, características o funcionalidades (*features*). Se centra en el usuario y no en el programador. Consta de 5 pasos o fases: *i)* realizar un modelo general del proyecto, *ii)* crear una lista de funcionalidades, *iii)* planear por funcionalidad, *iv)* diseñar por funcionalidad y *v)* construir por funcionalidad, las dos últimas fases son iterativas [21], [47], [48]. Véase una representación en la figura 14.

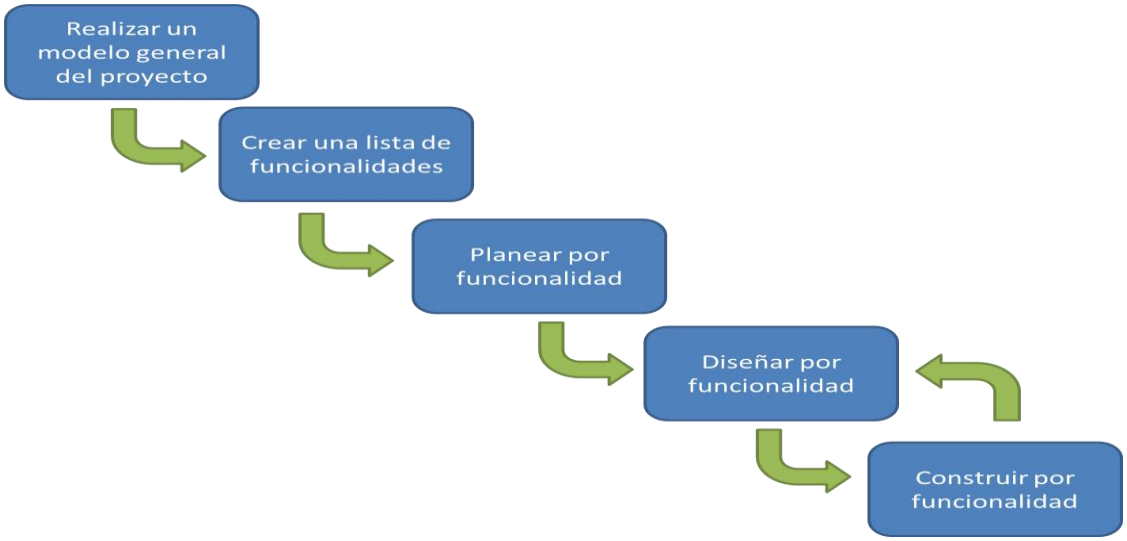


Figura 14. Proceso FDD. Adaptado de [47], [48].

5.8 Desempeño Computacional

5.8.1 Speed Up

Es un valor de medida numérico que indica el grado de ganancia o mejora en tiempos de respuesta en una aplicación comparándolo con una aplicación base diferente. Para ello, se toma el valor del tiempo requerido, tanto por la aplicación base como por aquella que se va a comparar en procesar determinada tarea para luego dividir el tiempo de la aplicación a comparar sobre el tiempo de la aplicación base.

$$S = \frac{t_1}{t_0} \tag{3}$$

Donde *S* es el *SpeedUp*, t_1 es el tiempo de la aplicación a comparar y t_0 es el tiempo de la aplicación base con la que se compara.

Si el valor es mayor a uno (>1) significa que hay una ganancia en tiempos de respuesta, por el contrario, si el valor es menor a uno (<1) hay pérdida y la nueva

aplicación se tarda más en dar resultados. Ahora, si el valor tiende a uno (->1) representa que no habrá cambios sustanciales en los tiempos.

$$\begin{gathered} S > 1 \rightarrow \ + \\ S < 1 \rightarrow \ - \\ S \cong 1 \rightarrow \ No\ hay\ cambio\ sustancial \end{gathered} \tag{4}$$

5.8.2 Ley de Amdahl

Para el análisis de rendimiento y eficiencia, la Ley de Amdahl se centra bastante en sistemas informáticos con procesos de ejecución en paralelo y/o que varían en determinada característica o condición comparable y medible con un estado base, para lo cual se implementó la siguiente versión de la ecuación

$$S = \frac{1}{(1-F_m)+\left(\frac{F_m}{A_m}\right)} \tag{5}$$

Donde F_m, es la fracción del tiempo del proceso mejorado o paralelizado que se está comparando y el A_m, es el factor de mejora de comparación [19].

6 Desarrollo de la metodología

6.1 Metodología de Investigación

Se implementó una metodología de investigación por fases secuenciales que estuvieron alineadas a los objetivos específicos representadas de forma general por la figura 15. En cada fase se tuvo como propósito desarrollar uno o varios de los objetivos específicos de este proyecto.

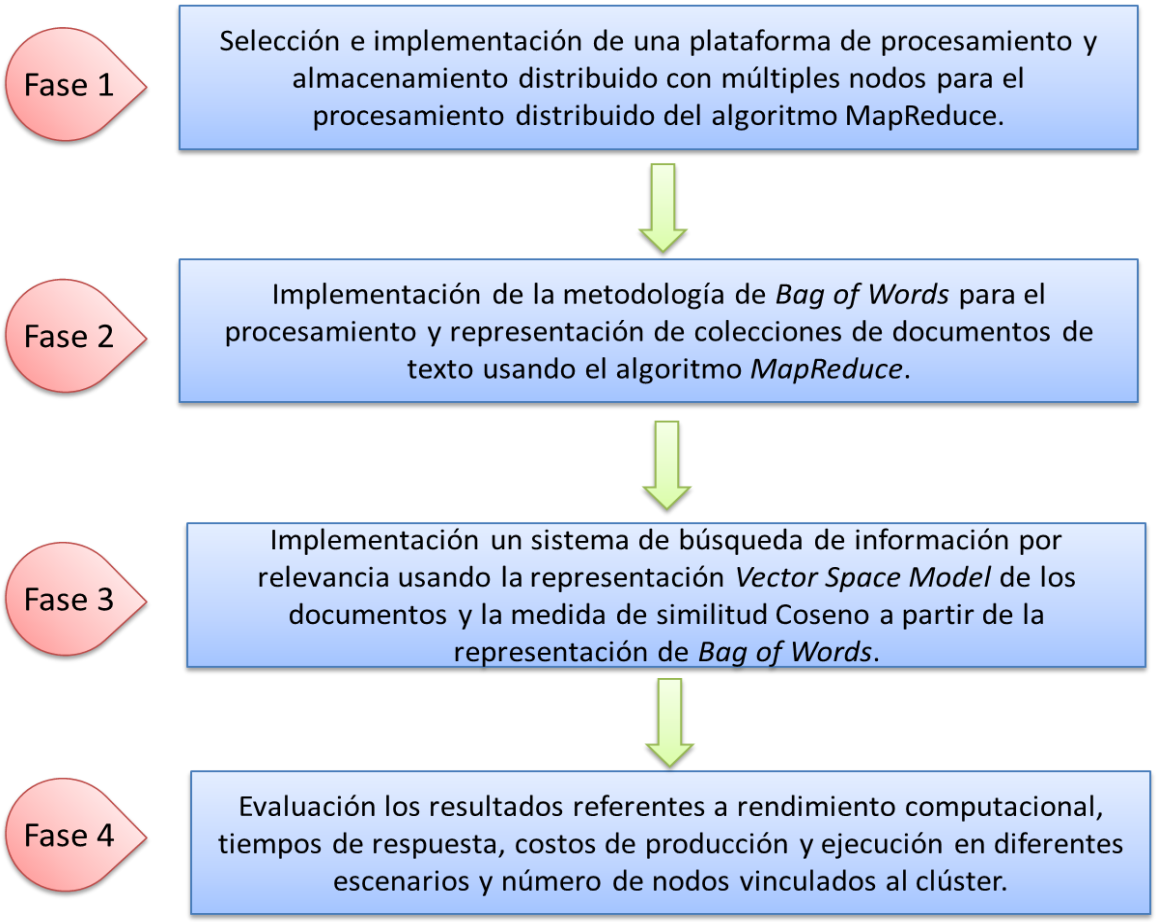


Figura 15. Metodología de Investigación. Fuente propia.

6.2 Metodología de Desarrollo de Software

Para el desarrollo de software se implementó FDD partiendo de una serie de requerimientos o funcionalidades definidas y relacionadas en la metodología de investigación.

6.2.1 Modelo General del proyecto

Se elaboró tres diagramas que encierran cada una de las funcionalidades y etapas de la aplicación y cómo se implementó en Apache Spark, ver figuras 16-18.

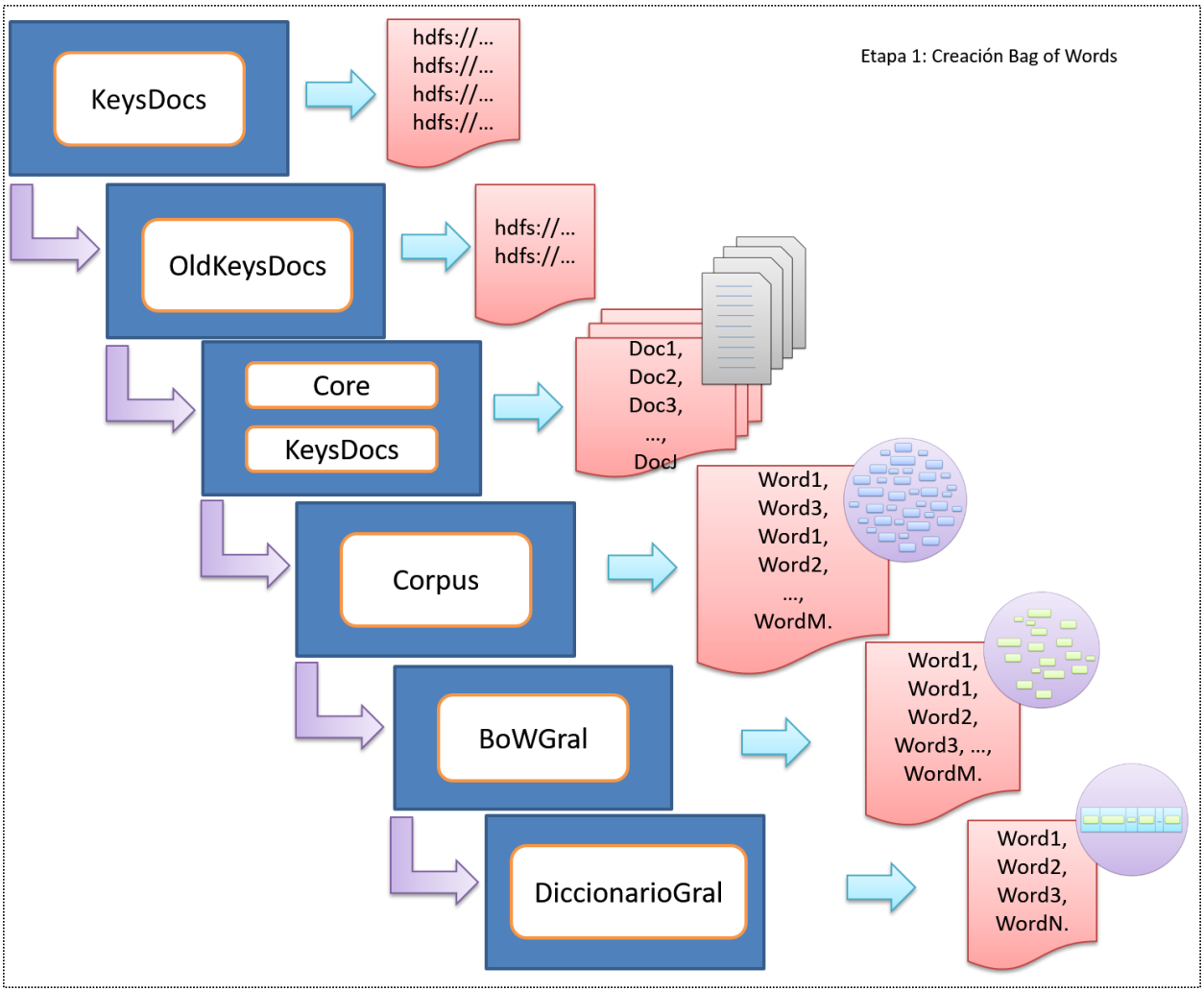


Figura 16. Diagrama de implementación de la representación de documentos, *Bag of Words*, Etapa 1: "Creación de *Bag of Words*" en Apache Spark. Fuente propia.

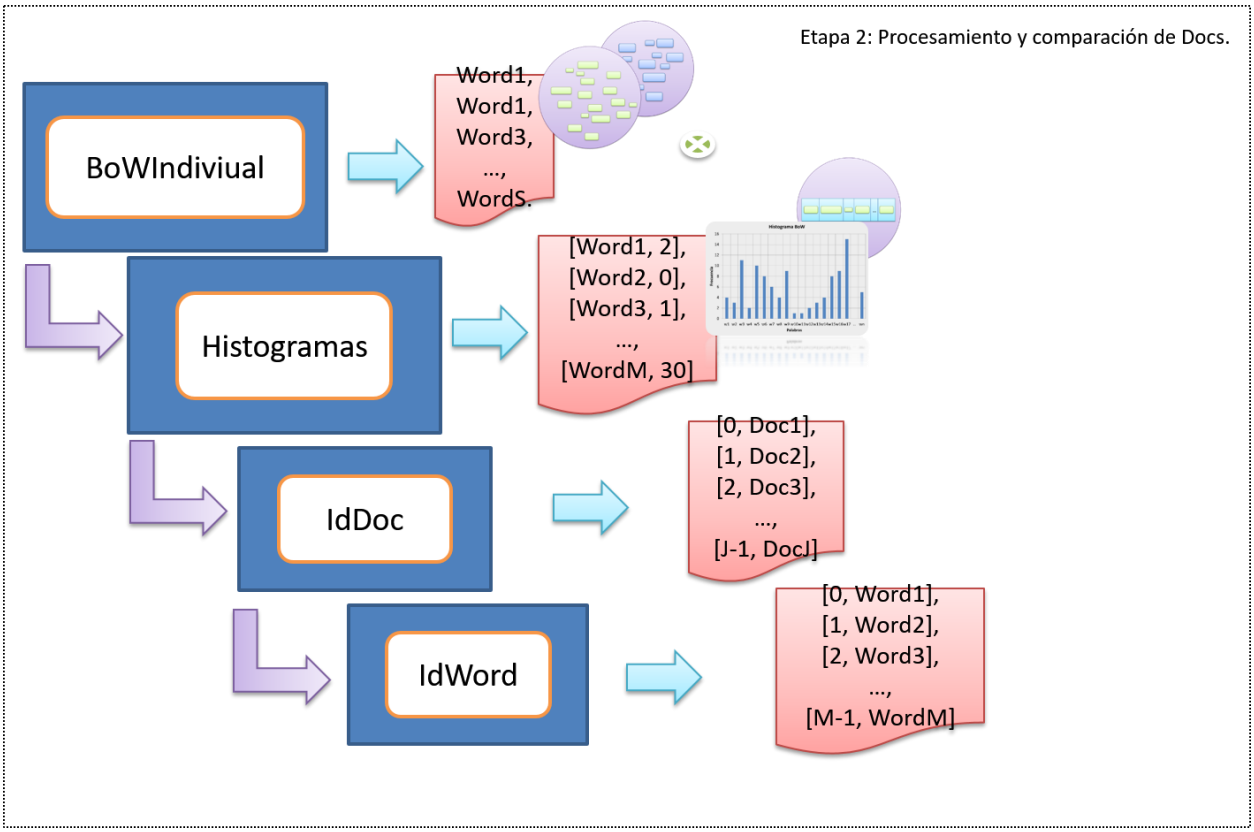


Figura 17. Diagrama de implementación de la representación de documentos, *Bag of Words*, Etapa 2: “Procesamiento y comparación de Docs” en Apache Spark. Fuente propia.

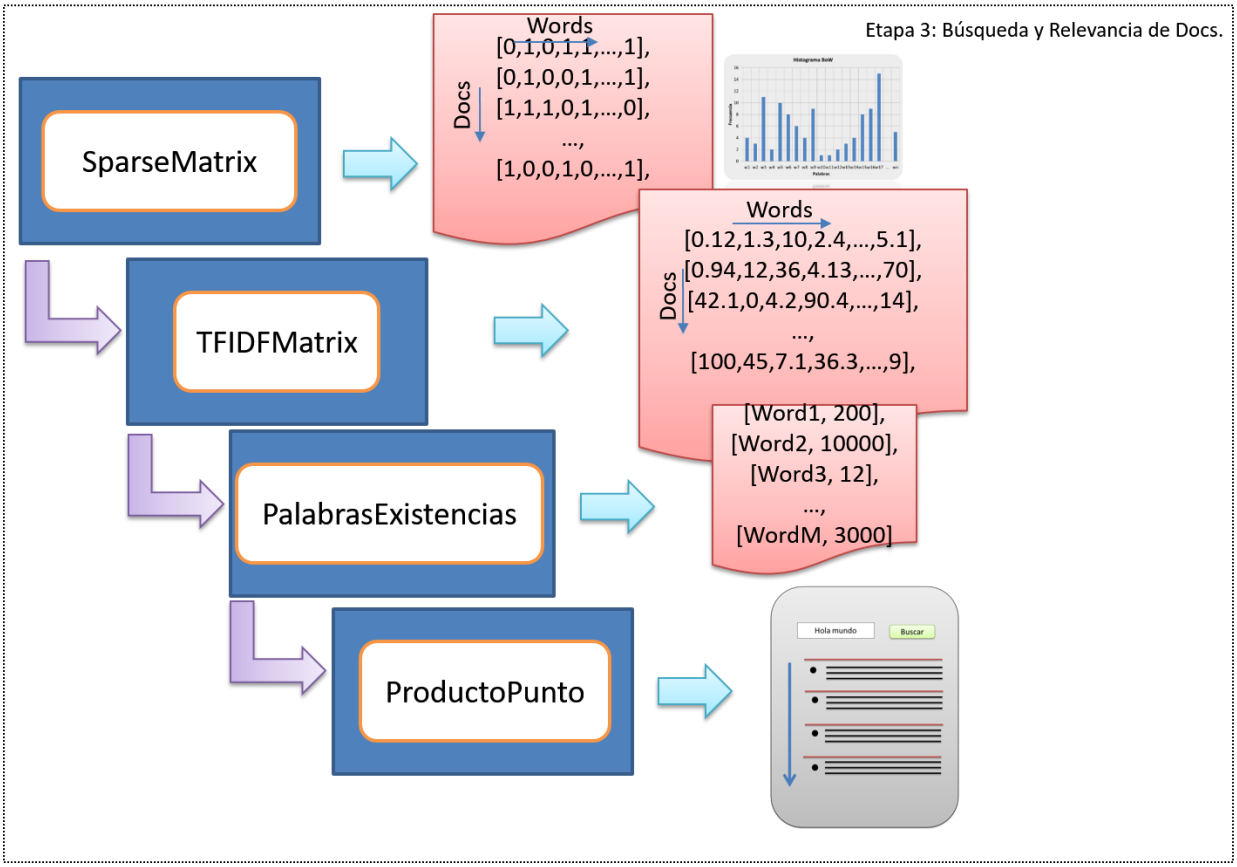


Figura 18. Diagrama de implementación de una aplicación de búsqueda y relevancia de palabras y documentos en Apache Spark. Fuente propia.

7 Aplicación Búsqueda de Documentos por Relevancia

A continuación, se explica la funcionalidad y estructura planteada que constituye la aplicación implementada en el clúster con las plataformas Apache Spark y Hadoop (HDFS). Anteriormente, en la Figura 16 (Diagrama general de la implementación) se representó una estructura general que mostraba cómo funcionaba la aplicación de búsqueda de documentos de texto por relevancia distinguiendo las fases y módulos que se plantearon realizar buscando aplicar el algoritmo de *Bag of Words*. Cada módulo se implementó por medio de un script cuyo resultado sería el entregable del

respectivo módulo, dicho script fue implementado en Apache Spark soportado sobre Python 2.7 a través de un Notebook de Jupyter[1].

7.1 Módulos de la Bolsa de Palabras

Como la idea no es repetir lo mencionado ya en la sección de la metodología de desarrollo de software, se dispone a describir puntualmente el concepto usado para cada etapa junto con los scripts creados

7.1.1 Etapa 1: Creación de Bag of Words

Consiste en procesar el dataset intacto, sin algún tipo de preprocesamiento, eliminando archivos inválidos que no consistan en texto plano coherente y con sentido literal para lo que se elimina todos los caracteres que no pertenezcan al alfabeto estado unidense (ya que el buscador se realizó para documentos en inglés) cuyo resultado final será un diccionario de términos.

7.1.2 Etapa 2: Procesamiento y Comparación de Documentos

Realizando de nuevo el procedimiento anterior para todos los documentos de forma separada para así obtener una representación *Bag of Words* específica para cada uno y luego contrastar con el diccionario de términos y finalizar con la creación del Histograma del *Bag of Words.*

7.1.3 Etapa 3: Búsqueda y Relevancia de Documentos.

[1] Jupyter: "*El cuaderno Notebook de IPython está basado en la web y es interactivo. Un notebook en IPython es un documento JSON que contiene una lista ordenada de entradas / salidas las cuales pueden tener código, texto, matemáticas, dibujos y rich media.*", Tomado de https://es.wikipedia.org/wiki/IPython. Página oficial: http://jupyter.org/

Al tener un archivo que indica la frecuencia de existencia de las palabras por documento, se continuó con el análisis y procesado de valores cuantitativos similares aplicando operaciones matemáticas pertinentes e. g. operación producto punto (ver ecuación 6), para ordenar por relevancia los resultados de la búsqueda realizada.

$$\boldsymbol{R} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{M}} \boldsymbol{W_{i,d}} * \boldsymbol{W_{i,q}} \tag{6}$$

7.2 Implementación BoW en Apache Spark

Para la explicación de los scripts creados, sólo se toman aquellos que se analizan en este libro más adelante, debido a que fueron considerados de mayor importancia sobre los demás.

7.2.1 Core (scriptCore.py)

Permite crear un Core de documentos limpios (sin caracteres especiales). Por cada documento almacenado en el HDFS se crea su homólogo limpio y se almacena también de forma distribuida.

7.2.2 Corpus (scriptCorpus.py)

Encargado de unir todos los documentos en un único archivo tomando todas sus palabras y uniéndolas con las demás para su posterior almacenamiento distribuido.

7.2.3 BoWGral (scriptBoWGral.py)

Crea el *Bag of Words* General a partir del Corpus y filtrando los *Stopwords*.

7.2.4 DiccionarioGral (scriptDiccionarioGral.py)

Script para la creación del diccionario general de palabras válidas unitarias que existen en todo el dataset.

7.2.5 BoWIndividual (scriptBoWIndividual.py)

Genera los *Bag of Words* para cada documento a partir de su respectivo corpus, para este caso el mismo core (siendo que el Core se hizo limpiando las palabras y creando un documento con el mismo nombre).

7.2.6 Histogramas (scriptHistogramas.py)

Representa por medio de un histograma cada *Bag of Words* de cada documento y lo almacena como un archivo tipo clave-valor llamado igual que el documento origen, en donde la clave es la palabra y el valor es la frecuencia.

7.2.7 idDoc (scriptIdDoc.py)

Script de rápida ejecución que busca crear un archivo tipo clave-valor identificando los documentos almacenados en el HDFS.

7.2.8 idWord (scriptIdWord.py)

Al igual que el anterior, crea un archivo tipo clave-valor identificando las palabras almacenadas en el HDFS.

7.2.9 SparseMatrix (scriptSparseMatrix.py)

Script con una carga en memoria considerable debido al procesamiento iterativo de los valores de los histogramas usados para crear la matriz binaria de existencias.

7.2.10 TFIDFMatrix (scriptTFIDFMatrix.py)

De igual forma que la *SparseMatrix,* requiere de bastante memoria para almacenar una cantidad considerable de datos y cálculos cuya carga al ser distribuida permite obtener valores que demuestran aumentos en tiempos de procesamiento al necesitar en repetidas ocasiones acceder a varios nodos por medio de la red. Su resultado final es una matriz de valores ponderados y normalizados.

7.2.11 Producto Punto (scriptProductoPunto.py)

En este parte, se desarrolló un código que permitiera usar la matrix TF-IDF generada por el script anterior par así realizar los correspondientes cálculos relacionados con el dataset y la consulta que también se hace en este script, para ya al finalizar dar unos resultados ordenados referentes a los documentos más relevantes según la búsqueda.

Es posible buscar por tres tipos de contenidos, por Paths (archivos locales), URL's (documentos web) y cadenas de texto plano.

8 Resultados

8.1 Conjunto de Datos

Se trabajó con un Dataset de 500 archivos de texto plano cuyo formato era irrelevante, no importaba si eran xml, txt y demás. El tamaño de los archivos oscilaba entre 50 KB y 2 MB aproximadamente. Se procuró optar por fuentes distintas con contenidos diferentes, de libre uso y distribución distribuidos de la siguiente manera:

1 - Reuters => 22 documentos [49].
2 - aclImdb => 100 documentos [50].

3 - 20_newsgroups => 120 documentos [51].

4 - Core.zip => 200 documentos [52]–[55].

5 - mix20_rand700_tokens_0211 => 58 documentos [56].

8.2 Diseño Experimental

Se realizaron en total cinco escenarios distintos de experimentación en donde cada uno representaba un clúster con una determinada cantidad propia de nodos. Los escenarios propuestos fueron para 1, 2, 4, 8 y 16 nodos. La medida de desempeño del clúster para evaluar el procesamiento del conjunto de datos fue el tiempo de computación (segundos) en que tardaban en culminar exitosamente cada uno de los módulos y etapas de la implementación en Apache Spark para obtener la representación de los documentos como Bag of Words.

Se procuró que todos los nodos contaran con las mismas características para un análisis más sencillo partiendo de un clúster con nodos homogéneos. Para ello se configuró una máquina virtualizada con las siguientes características:

- Sistema Operativo Ubuntu 14.04.5 LTS.
    - Arquitectura: 64 bit.
    - Versión: Trusty Tahr.
- Memoria RAM:
    - Nodo Maestro: 4096 MB (4 GB).
    - Nodos Esclavos (x16): 14336 MB (14 GB).
- Almacenamiento: Disco Duro Sata de 30 GB.
- Unidad de Procesamiento: 2 Core.
- Red: Adaptador puente. Se configuró la red con las asignaciones de IP, mostradas en la tabla 1.

Tabla 1. Estructura de Red. Asignación de IPs de los diferentes nodos del Clúster.

IP	Hostname
192.168.1.2	Master #Coordinador
192.168.1.4	Worker1 #Slave1
192.168.1.5	Worker2 #Slave2
192.168.1.6	Worker3 #Slave3
192.168.1.7	Worker4 #Slave4
192.168.1.8	Worker5 #Slave5
192.168.1.9	Worker6 #Slave6
192.168.1.10	Worker7 #Slave7
192.168.1.11	Worker8 #Slave8
192.168.1.12	Worker9 #Slave9
192.168.1.13	Worker10 #Slave10
192.168.1.14	Worker11 #Slave11
192.168.1.15	Worker12 #Slave12
192.168.1.16	Worker13 #Slave13
192.168.1.17	Worker14 #Slave14
192.168.1.18	Worker15 #Slave15
192.168.1.19	Worker16 #Slave16

Cada módulo de la Figura 16 hace referencia, por lo general, a una tarea y/o una fase de la implementación del algoritmo *Bag of Words* en Apache Spark, los cuales se midieron aquellos que presentaran características destacables y que se consideraban de mayor carga de procesamiento, memoria y comunicación, siendo denominadas de alto impacto en el desempeño computacional para el clúster. En total fueron once los módulos de la implementación de *Bag of Words* en Apache Spark para su análisis y estudio correspondiente, los cuales se describen a continuación:

- Core
- Corpus
- BoWIndividual
- Histogramas
- TFIDFMatrix
- ProductoPunto, constituye el Buscador.

Para el Buscador se realizó la misma consulta en cada uno de los escenarios, una consulta por cadena de texto y una por un documento.

- Consulta por texto: *“home abandoned food official races”*
- Consulta por documento: en el repositorio del Proyecto (Véase la sección “Anexos y Repositorios”) se encuentra ubicado en “Proyecto de Grado / Documento_Consulta / otrodoc.txt”

Cada módulo se ejecutó tres veces con el fin de analizar la variabilidad del tiempo de ejecución, estimar la tendencia central y su dispersión, Algunos módulos se ejecutaron en menor cantidad al ser considerados, evaluados y aceptados debido a la naturaleza de su funcionamiento y codificación que ocasionaba que fueran señalados de simples y/o que su tiempo requerido para terminar su proceso era demasiado extenso y los tiempos en los cuales variaba no eran significativos.

En total se implementaron once módulos, y cada uno ejecutado para cinco escenarios, donde cada combinación de módulo y escenario se ejecutó en promedio tres veces, lo que resulta en un total de 165 tareas ejecutadas aproximadamente, las cuales se analizaron.

Posteriormente, tomando los tiempos de ejecución de cada módulo se hallaron los valores de aceleración *SpeedUp* experimentales y por medio de una gráfica de dispersión se compararon con los *SpeedUp* teóricos dando resultados variables tomando el módulo más similar demostrando los aumentos en aceleración indicando así el nivel de paralelización relacionado.

8.3 Resultados preliminares

Al realizar el proceso de ejecución y toma de datos se tomaron dos valores de dos fuentes distintas que permitían corroborar la veracidad de los mismos, uno era

generado por el mismo Notebook de Jupyter al usar el comando `%run` previo al llamado de un determinado script, por ejemplo:

```
%time %run scriptKeysDocs.py
```

El cual permitía obtener al ejecutar cada módulo el tiempo de ejecución del mismo. Por otro lado, la interfaz de usuario de Apache Spark proveía de un registro de las aplicaciones que ejecutaba y el tiempo de ejecución que tardaban. No obstante, se observó que este último valor era contantemente redondeado perdiendo credibilidad y confiabilidad por lo que, al final, sólo se consideró el valor de tiempo de ejecución obtenido directamente en el Notebook de Jupyter.

La tabla 2 se presentan los tiempos de ejecución en segundos (s) para cada uno de los módulos (Core, Corpus, BoWGral, DiccionarioGral, BoWIndividual, Histogramas, idDoc, idWord, SparseMatrix, TFIDFMatrix y ProductoPunto) para las tres ejecuciones por cada número de nodos (1, 2, 4, 8 y 16), su respectivo promedio, tanto para los tiempos registrados por la interfaz de Notebook Jupyter como directamente de Spark.

En algunos casos, la herramienta de Apache Spark arrojó valores decimales en minutos o incluso en horas, con una sola cifra de precisión por lo que al hacer la conversión del tiempo dado a segundos se perdía exactitud.

Tabla 2. Tiempos de ejecución de los módulos según las herramientas de Jupyter y Apache Spak.

	TIEMPO JUPYTER (%RUN)				TIEMPO SPARK			
Experimento 1 - 1 nodo								
Tareas\Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)
Core	1189	1247	1244	1226,667	1200	1260	1260	1240
Corpus	179	177	176	177,333	180	180	174	178
BowGral	185	185	183	184,333	186	186	180	184

DiccionarioGral	17,5	17,8	18,3	17,867	16	17	17	16,667
BoWIndividual	325	320	320	321,667	324	318	318	320
Histogramas	758	701	704	721	780	720	720	740
idDoc	6,76	6,04	6,66	6,4867	6	5	5	5,333
idWord	5,72	7,25	5,29	6,0867	5	6	4	5
SparseMatrix	611	637	627	625	660	660	600	640
TFIDFMatrix	18065	-	-	18065	18000	-	-	18000
ProductoPunto	193,75	174,75	-	184,25	193,5	-	-	193,5
Experimento 2 - 2 nodos								
Tareas\Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)
Core	1217	1217	1238	1224	1200	1200	1260	1220
Corpus	106	110	105	107	108	108	102	106
BowGral	137	138	139	138	138	138	138	138
DiccionarioGral	17,5	17,7	17,8	17,667	16	17	17	16,667
BoWIndividual	269	272	269	270	270	270	270	270
Histogramas	630	624	617	623,667	600	600	600	600
idDoc	9,15	6,63	6,75	7,51	8	6	6	6,667
idWord	7,21	6,05	6,77	6,677	4	5	6	5
SparseMatrix	620	619	622	620,333	600	600	600	600
TFIDFMatrix	-	-	-	-	-	-	-	-
ProductoPunto	-	-	-	-	-	-	-	-
Experimento 3 - 4 nodos								
Tareas\Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)
Core	1272	1248	1299	1273	1260	1260	1320	1280
Corpus	81	89	91	87	84	90	90	88
BowGral	124	124	125	124,333	126	126	126	126
DiccionarioGral	19,2	18,5	19,9	19,2	18	17	19	18

BoWIndividual	306	296	296	299,333	306	294	294	298
Histogramas	729	684	698	703,667	720	660	720	700
idDoc	10,1	10,7	10,2	10,333	9	8	9	8,667
idWord	9,21	7,91	7,2	8,1067	7	7	6	6,667
SparseMatrix	662	658	663	661	660	660	660	660
TFIDFMatrix	-	-	-	-	-	-	-	-
ProductoPunto	-	-	-	-	-	-	-	-
Experimento 4 - 8 nodos								
Tareas\Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)
Core	1322	1360	1354	1345,333	1320	1380	1380	1360
Corpus	62	62	62	62	62	60	60	60,667
BowGral	124	104	112	113,333	126	102	114	114
DiccionarioGral	23,3	23,9	27,9	25,0333	22	23	27	24
BoWIndividual	364	362	366	364	366	360	366	364
Histogramas	869	889	880	879,333	840	900	900	880
idDoc	13,9	10,9	10,9	11,9	13	10	10	11
idWord	8,05	8,56	6,89	7,8333	7	7	6	6,667
SparseMatrix	936	922	919	925,667	960	900	900	920
TFIDFMatrix	18134	-	-	18134	18000	-	-	18000
ProductoPunto	283	217,5	-	250,25	282	214,5	-	248,25
Experimento 5 - 16 nodos								
Tareas\Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)
Core	1400	-	-	1400	1380	-	-	1380
Corpus	49,9	43	41,1	44,667	46	42	41	43
BowGral	114	111	108	111	114	108	108	110
DiccionarioGral	29,8	19,1	29,5	26,133	29	18	28	25
BoWIndividual	396	394	402	397,333	396	396	402	398

Histogramas	1099	1111	1271	1160,333	1080	1140	1260	1160
idDoc	17,7	16,1	17,6	17,1333	17	15	16	16
idWord	11,2	11,6	11,6	11,4667	10	10	10	10
SparseMatrix	1021	1017	993	1010,333	1020	1020	1020	1020
TFIDFMatrix	18167	-	-	18167	18360	-	-	18360
ProductoPunto	247,5	222	-	234,75	246	222	-	234

8.4 Resultados finales

Al finalizar la selección y filtrado de resultados relevantes se dispuso a la creación de nuevas tablas con valores que permitieran el correcto análisis y representación de los tiempos de ejecución y los *Speed Up* para cada caso. El resto de la experimentación se realizó únicamente con los tiempos de ejecución obtenidos del Notebook Jupyter, descartando los tiempos de la herramienta de Apache Spark. Para analizar el grado de dispersión, se calculó la desviación estándar de cada medición por experimento (número de nodos) y módulo a partir de las tres repeticiones de ejecución cuyos resultados se pueden relacionar en la tabla 3.

Tabla 3. Tiempos de ejecución y desviación estándar de los módulos según la herramienta de Jupyter.

TIEMPO JUPYTER (%RUN)					
Experimento 1 - 1 nodo					
Tareas\ Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	Promedio (s)	Desviación Estándar
Core	1189	1247	1244	1226.667	32.655
Corpus	179	177	176	177.333	1.528
BowGral	185	185	183	184.333	1.155
DiccionarioGral	17.5	17.8	18.3	17.867	0.404
BoWIndividual	325	320	320	321.667	2.887
Histogramas	758	701	704	721.000	32.078
idDoc	6.76	6.04	6.66	6.487	0.390

idWord	5.72	7.25	5.29	6.087	1.030
SparseMatrix	611	637	627	625.000	13.115
TFIDFMatrix	18065	-	-	18065.000	0
ProductoPunto	193.75	174.75	-	184.250	13.435
Experimento 2 - 2 nodos					
Tareas\ Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	FINAL	Desviación Estándar
Core	1217	1217	1238	1224.000	12.124
Corpus	106	110	105	107.000	2.646
BowGral	137	138	139	138.000	1.000
DiccionarioGral	17.5	17.7	17.8	17.667	0.153
BoWIndividual	269	272	269	270.000	1.732
Histogramas	630	624	617	623.667	6.506
idDoc	9.15	6.63	6.75	7.510	1.422
idWord	7.21	6.05	6.77	6.677	0.586
SparseMatrix	620	619	622	620.333	1.528
TFIDFMatrix	-	-	-	0	0
ProductoPunto	-	-	-	0	0
Experimento 3 - 4 nodos					
Tareas\ Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	FINAL	Desviación Estándar
Core	1272	1248	1299	1273.000	25.515
Corpus	81	89	91	87.000	5.292
BowGral	124	124	125	124.333	0.577
DiccionarioGral	19.2	18.5	19.9	19.200	0.700
BoWIndividual	306	296	296	299.333	5.774
Histogramas	729	684	698	703.667	23.029
idDoc	10.1	10.7	10.2	10.333	0.321
idWord	9.21	7.91	7.2	8.107	1.019
SparseMatrix	662	658	663	661.000	2.646
TFIDFMatrix	-	-	-	0	0
ProductoPunto	-	-	-	0	0
Experimento 4 - 8 nodos					
Tareas\ Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	FINAL	Desviación Estándar
Core	1322	1360	1354	1345.333	20.429
Corpus	62	62	62	62.000	0.000
BowGral	124	104	112	113.333	10.066
DiccionarioGral	23.3	23.9	27.9	25.033	2.501
BoWIndividual	364	362	366	364.000	2.000
Histogramas	869	889	880	879.333	10.017
idDoc	13.9	10.9	10.9	11.900	1.732
idWord	8.05	8.56	6.89	7.833	0.856
SparseMatrix	936	922	919	925.667	9.074
TFIDFMatrix	18134	-	-	18134.000	0
ProductoPunto	283	217.5	-	250.250	46.315

Experimento 5 - 16 nodos					
Tareas\ Tiempos	Ejecución 1 (s)	Ejecución 2 (s)	Ejecución 3 (s)	FINAL	Desviación Estándar
Core	1400	1390	1420	1403.333	15.275
Corpus	49.9	43	41.1	44.667	4.631
BowGral	114	111	108	111.000	3.000
DiccionarioGral	29.8	19.1	29.5	26.133	6.093
BoWIndividual	396	394	402	397.333	4.163
Histogramas	1099	1111	1271	1160.333	96.028
idDoc	17.7	16.1	17.6	17.133	0.896
idWord	11.2	11.6	11.6	11.467	0.231
SparseMatrix	1021	1017	993	1010.333	15.144
TFIDFMatrix	18167	-	-	18167.000	0
ProductoPunto	247.5	222	-	234.750	18.031

Para el caso del módulo del buscador, el cual implementa el producto punto para comparar los documentos del corpus en la consulta (query), `ProductoPunto`, se extrajo cuatro diferentes tiempos de respuesta, el primer valor consiste en el tiempo general dado por el Notebook de Jupyter, para los otros tres, se usó la función `time()` de Python y así dividir e indicar los demás tiempos: lo que tarda en operar la consulta y buscar, en imprimir los resultados y el total invertido en este proceso, siendo la suma de los dos mencionados anteriormente. Este proceso se hizo únicamente con el fin de indagar en los tiempos que conlleva el buscador al tener ya todas las fases del *Bag of Word* previamente cargadas, almacenadas y listas para su uso, por lo que sólo se tomaron datos de dos escenarios, con uno y ocho nodos, respectivamente. Obsérvese el resumen en las tablas 4 y 5.

Tabla 4. Tiempos de ejecución. Módulo Buscador – 1 nodo.

1 NODO			
Cadena		Path	
Gral (s)	Time() (s)	Gral (s)	Time() (s)
198	122.22	177	125.53
	50.23		31.9
	179.46		157.44
191	124.33	175	124.98
	48.54		31.44
	172.88		156.42

191	125.44	171	120.85
	46.87		30.59
	172.32		151.45
195	125.87	176	125.78
	49.35		32.02
	175.22		157.81

Tabla 5. Tiempos de ejecución. Módulo Buscador – 8 nodos.

8 NODOS			
Cadena		Path	
260	153.78	223	136
	71.1		34
	234.7		170.8
301	187.2	233	162.2
	82.45		44.7
	269.68		206.9
264	159.59	206	141
	76.12		38
	235.71		179.2
307	183.75	208	147.2
	83.59		36.5
	267.34		183.79

Usando los tiempos dados por el Notebook en Jupyter, se resumió en tablas de tal manera que fuera más fácil su manejo y comprensión. En primer lugar, los tiempos (en segundos) de ejecución de cada módulo con su respectivo escenario se muestran en la tabla6.

Tabla 6. Tiempos de ejecución de los módulos. Resumen.

Nodos\Tareas.py	Core	Corpus	BoWGral	DiccionarioGral	BoWIndividual	Histogramas	idDoc	idWord	SparseMatrix	TFIDFMatrix	Producto Punto
1	1226.667	177.333	184.333	17.867	321.667	721.000	6.487	6.087	625.000	18065.000	184.250
2	1224.000	107.000	138.000	17.667	270.000	623.667	7.510	6.677	620.333	-	-
4	1273.000	87.000	124.333	19.200	299.333	703.667	10.333	8.107	661.000	-	-
8	1345.333	62.000	113.333	25.033	364.000	879.333	11.900	7.833	925.667	18134.000	250.250

16	1403.333	44.667	111.000	26.133	397.333	1160.333	17.133	11.467	1010.333	18167.000	234.750

Posteriormente, la tabla 7 contiene las desviaciones estándar de cada módulo señalando así el grado de dispersión de las tres ejecuciones realizadas.

Tabla 7. Desviaciones Estándar de los tiempos de ejecución de los módulos.

Nodos\ Módulos	Core	Corpus	BoWGral	DiccionarioGral	BoWIndividual	Histogramas	idDoc	idWord	Sparse Matrix	TFIDF Matrix	ProductoPunto
1	32.655	1.528	1.155	0.404	2.887	32.078	0.390	1.030	13.115	0.000	13.435
2	12.124	2.646	1.000	0.153	1.732	6.506	1.422	0.586	1.528	-	-
4	25.515	5.292	0.577	0.700	5.774	23.029	0.321	1.019	2.646	-	-
8	20.429	0.000	10.066	2.501	2.000	10.017	1.732	0.856	9.074	0.000	46.315
16	15.275	4.631	3.000	6.093	4.163	96.028	0.896	0.231	15.144	0.000	18.031

Por último, la tabla 8 muestra el aumento porcentual del rendimiento de todos los datos en sus escenarios, la tabla del *Speed Up*.

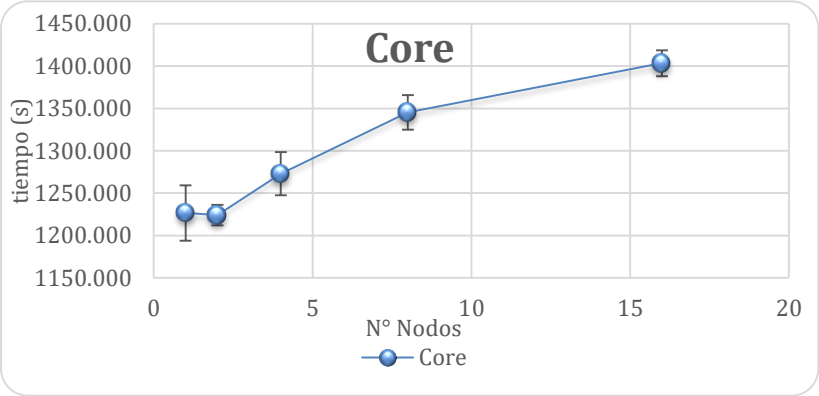
Tabla 8. Valores *SpeedUp* de los módulos de ejecución.

Nodos\ Módulos	Core	Corpus	BoWGral	DiccionarioGral	BoWIndividual	Histogramas	idDoc	idWord	Sparse Matrix	TFIDF Matrix	ProductoPunto
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.002	1.657	1.336	1.011	1.191	1.156	0.864	0.912	1.008	-	-
4	0.964	2.038	1.483	0.931	1.075	1.025	0.628	0.751	0.946	-	-
8	0.912	2.860	1.626	0.714	0.884	0.820	0.545	0.777	0.675	0.996	0.736
16	0.874	3.970	1.661	0.684	0.810	0.621	0.379	0.531	0.619	0.994	0.785

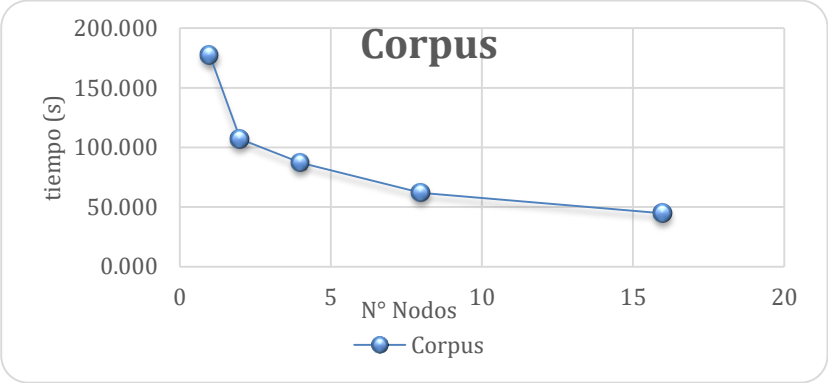
Si bien, las tablas permiten un mejor entendimiento, al ser bastantes datos y requerir que se permita observar un comportamiento según las variables evaluadas, es más que obligatorio graficar y generar una función teórica que permita linealizar y prever comportamientos y relaciones en diferentes escenarios a mayor escala.

Primero, se observan las gráficas 1-11 basadas en los datos recopilados de los diferentes tiempos resultantes de las ejecuciones de los módulos por medio de una línea de color azul. Además, se incluyó un elemento gráfico para visualizar las

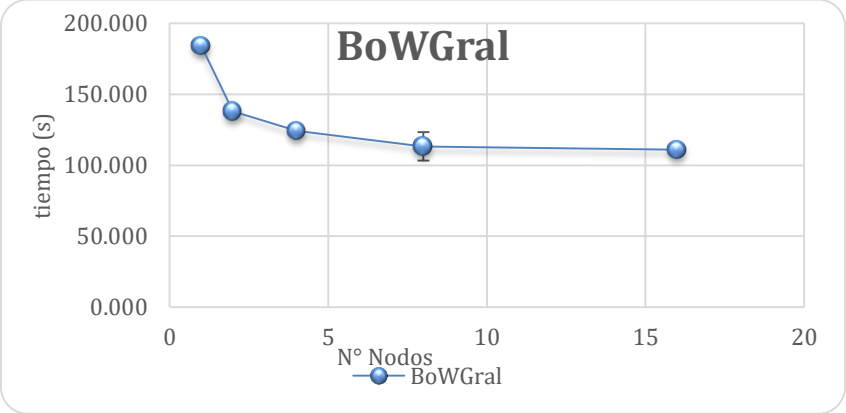
desviaciones estándar de cada punto en la función, las líneas verticales de color negro que se encuentran en cada punto.



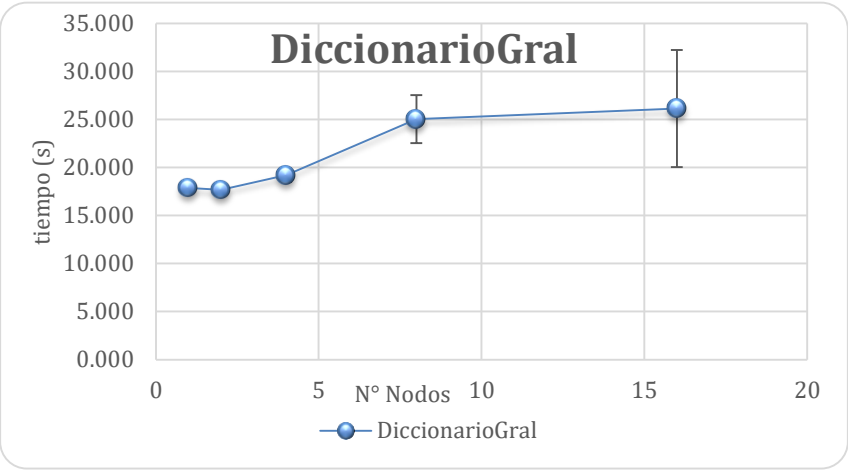
Gráfica 1. Tiempos de ejecución. Módulo Core.



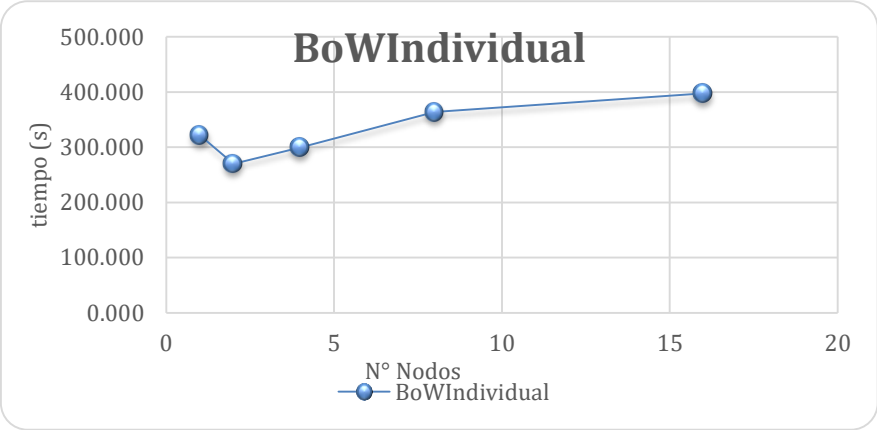
Gráfica 2. Tiempos de ejecución. Módulo Corpus.



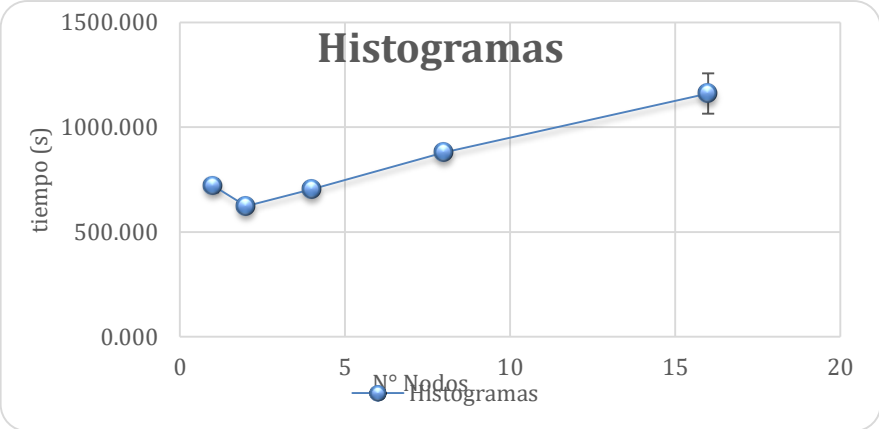
Gráfica 3. Tiempos de ejecución. Módulo BoWGral.



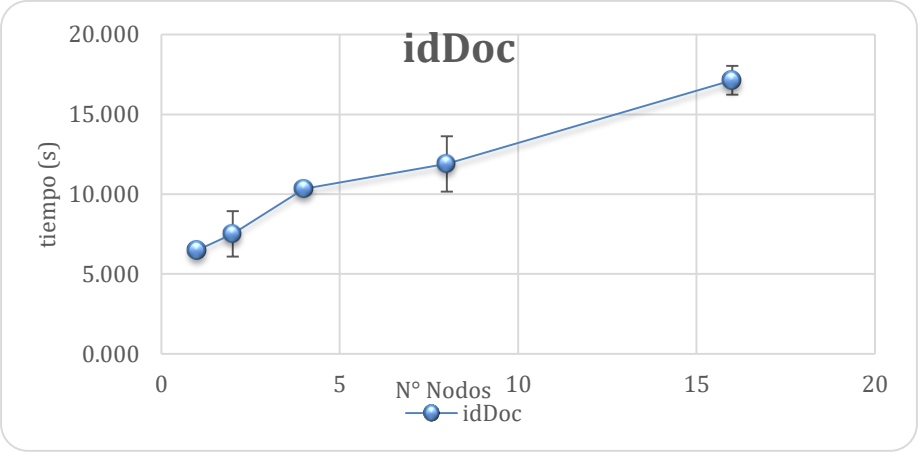
Gráfica 4. Tiempos de ejecución. Módulo DiccionarioGral.



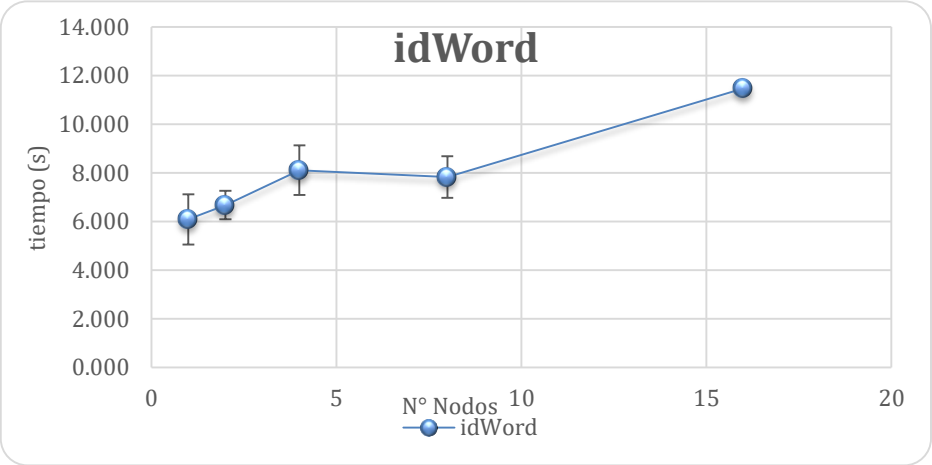
Gráfica 5. Tiempos de ejecución. Módulo BoWIndividual.



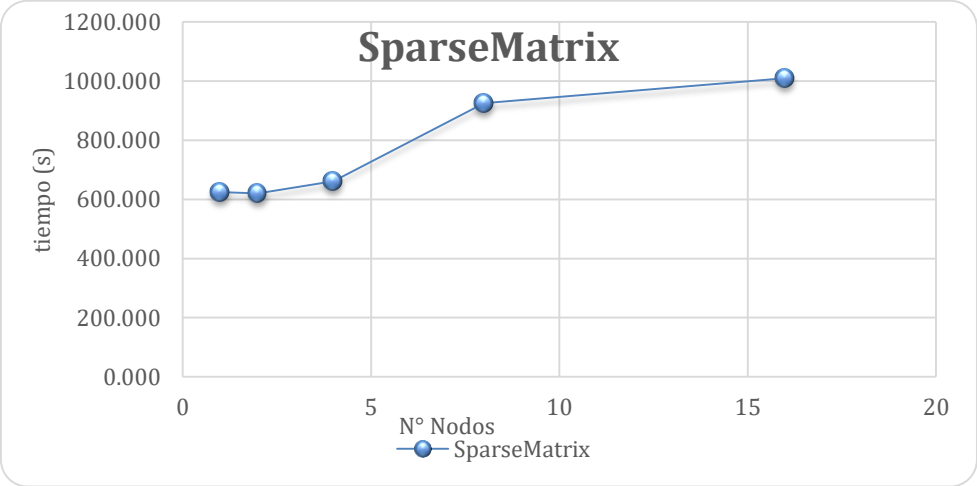
Gráfica 6. Tiempos de ejecución. Módulo Histogramas.



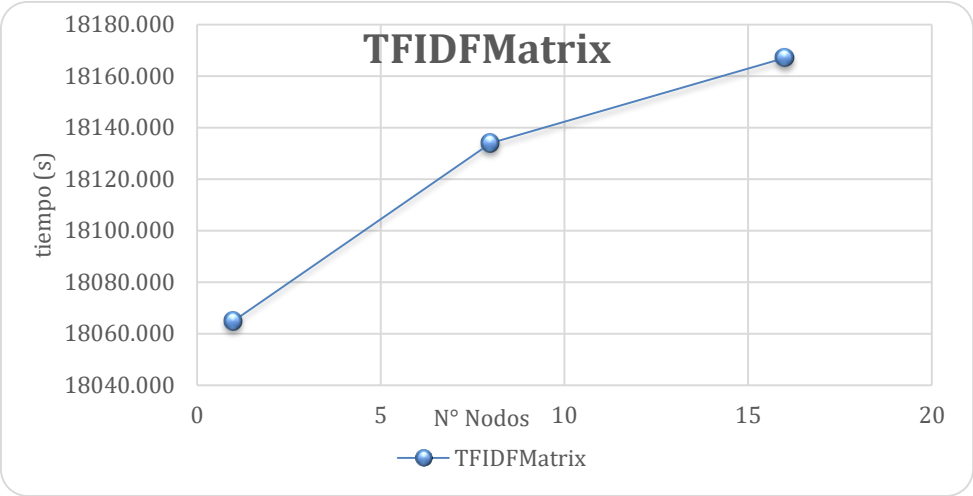
Gráfica 7. Tiempos de ejecución. Módulo idDoc.



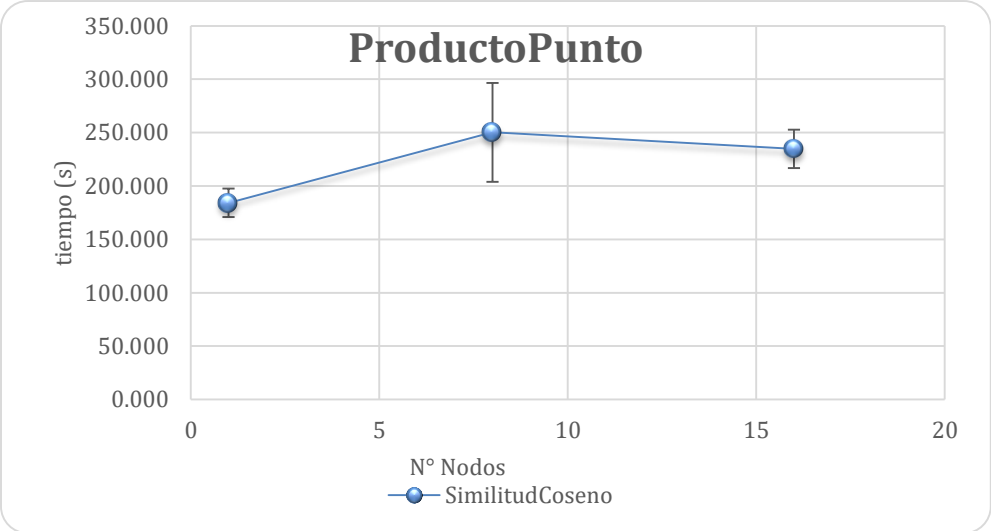
Gráfica 8. Tiempos de ejecución. Módulo idWord.



Gráfica 9. Tiempos de ejecución. Módulo SparseMatrix.

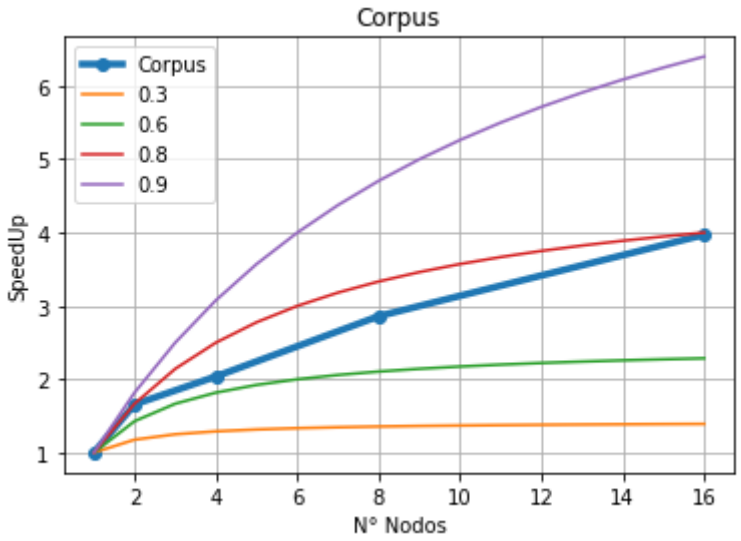


Gráfica 10. Tiempos de ejecución. Módulo TFIDFMatrix.



Gráfica 11. Tiempos de ejecución. Módulo ProductoPunto.

Posteriormente, en la gráfica 12 se representaron los resultados obtenidos por la ejecución del módulo **Corpus** cuyo comportamiento permite relacionar el *SpeedUp* con lo planteado en este trabajo y así predecir un punto de estabilidad demostrando el nivel de paralelización de los procesos del clúster en relación con la cantidad de nodos que se implementan para ello.



Gráfica 12. *SpeedUp* Teóricos y *SpeedUp* Módulo Corpus.

8.5 Análisis de Resultados

8.5.1 Desviación Estándar

Como se pudo observar tanto en las tablas como en las gráficas, la mayoría de puntos o muestras poseen poca dispersión referente al valor medio tomado como representativo. Permitiendo considerar que Apache Spark ocupa casi los mismos recursos y tiempos de ejecución por cada tarea que se le asigna para realizar en reiteradas ocasiones.

Cabe señalar que en el momento de la experimentación se notó que cuando se cargaba la plataforma y se ejecutaba una tarea por primera vez, dicho proceso tendía a demorarse un poco más a las siguientes que se hicieran siendo la misma. Esto se ve en las tablas de tiempos de ejecución para ciertas tomas que son un tanto diferentes a las otras 2. Se dejaron estos valores con intención de demostrar para análisis este

comportamiento, demostrando que Spark almacena en memoria algunos datos mejorando los tiempos de respuesta de procesos que los requieran con frecuencia.

8.5.2 *SpeedUp* Descendente

Si bien, se pretendía en un inicio que las gráficas del *SpeedUp* fueran logarítmicas positivas, se observó en muchas de las gráficas, todo lo contrario. Un comportamiento totalmente descendente era tendencia indicando de esta forma que cada vez era más lento el proceso en dar resultados.

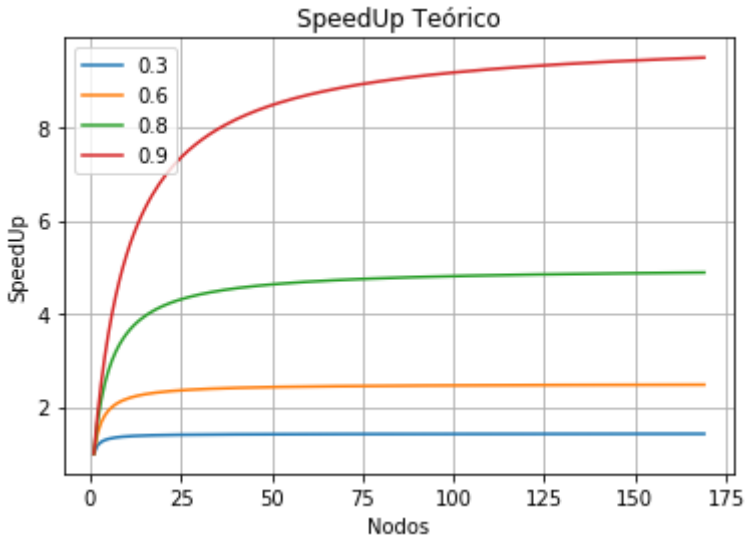
Luego de investigar cuidadosamente las causas de esta disminución, evaluando conexiones de comunicación físicas, hardware, software, scripts y demás. Se plantearon algunas hipótesis que se fueron corroborando en cada escenario experimental que se realizaba demostrando un patrón similar, siendo la comunicación a través de la red el causante. En otras palabras, el proceso experimental se realizó en 17 máquinas (16 nodos esclavos y un nodo maestro), no obstante, eran en total 32 equipos los conectados directamente al mismo Router que servía de medio y puente de comunicación, por lo que cada equipo físico contaba con un sistema operativo virtual usado para este trabajo experimental por lo que se consideran en esos casos, 2 sistemas operativos diferentes conectados y transmitiendo datos en red.

Otro causal relacionado con la comunicación, son los scripts que se ejecutaban en la plataforma. Dichos scripts presentaban una característica en común, realizaban muchísimos cálculos enviando y recibiendo datos por la red, debido a que algunas funciones de lectura de Spark conllevaban segundos de ejecución y se realizaban en reiteradas ocasiones. Ocasionando que los tiempos de respuesta aumentaran.

8.5.3 *SpeedUp* Ascendente

Scripts cuyas tareas se habían planteado de tal forma que realizaran poca comunicación en red y en cambio, realizaran más procesamiento en cada uno de los

nodos y repartieran mejor las cargas operacionales, dieron resultados muy similares a los teóricos esperados con comportamiento logarítmico positivo permitiendo relacionarse directamente con la Ley de Amdahl y la Gráfica de *SpeepUd* teórica. Para tal caso se tomó el script Corpus y como se demostró en la **Gráfica 12**, tenía un aumento o aceleración comparable a un 80% de paralelización de procesos en el cluster por lo que si tomamos la gráfica teórica de la *SpeedUp* para la función de paralelización del 80% y desplegamos a un número mayor de nodos (eje x) sabremos hasta qué punto se supone un aumento en la aceleración y por tanto, se recomienda la adquisición de equipos, implementación e inclusión al clúster cuyo objetivo sea desarrollar aplicaciones que presenten y requieran características planteadas en este trabajo cuya estructura y sentido de programación tenga el comportamiento del ya mencionado script Corpus.



Gráfica 13. SpeedUp Teórico.

9 Conclusiones

A pesar que los tiempos de respuesta son relativamente altos comparados con buscadores reconocidos, e. g. Google o Bing, nos permiten entender el funcionamiento y la eficiencia que se ofrece actualmente y pueden llegar a ofrecer con algo de trabajo, investigación y mejores equipos de cómputo.

A manera de estudio e investigación posterior, se propone indagar más acerca del cuestionamiento planteado en la sección SpeedUp Descendente en busca de posibles mejoras ya sea estructurando mejor la arquitectura de la red, mejorando los scripts de tal forma que sean más óptimos y mejor orientados al procesamiento y almacenamiento distribuido de *Big Data*.

10 Recomendaciones

Tener muy presente las observaciones hechas en este trabajo cuando se pretenta instalar, configurar e implementar las plataformas de almacenamiento y procesamiento distribuido como los son HDFS y Apache Spark, ya que incluso en este trabajo tuvieron que hacer bastantes cambios a la configuración tanto de esas plataformas como de los equipos usados para alojarlos (nodos).

Un claro ejemplo de esto fue el aumento en el tamaño del buffer usado por Java para la transmisión de datos por la red debido a que se requería más capacidad para ejecutar correctamente el script TFIDFMatrix causado por la gran cantidad de consultas y operaciones entre nodos en repetidas ocasiones.

11 Anexos y Repositorios

Toda la configuración, y de por sí las plataformas, están disponibles al público en repositorios Git para su libre estudio e implementación bajo sus propias responsabilidades a lo que se alojaron todos los archivos en BitBucket en 3 repositorios diferentes:

11.1 Plataforma Proyecto:

Contiene esencialmente 3 cosas: los scripts usados en este proyecto de investigación siendo todos creados en Python (.py), los Notebook de Jupyter que se iban trabajando buscando un mejor manejo del espacio en la interfaz web, IMPORTANTE señalar que la interfaz final usada en este trabajo y que usa correctamente todos los scripts es la que lleva por nombre **Corpus.ipynb** y por último contiene un archivo que cuenta con el dataset usado y además con una copia de todos los entregables y archivos resultantes de los scripts (e. g. Matriz TFIDF, Matriz Dispersa, Archivos Filtrados y limpios, etc.).

Enlace a repositorio Git: https://bitbucket.org/Yerfer/proyecto-de-grado/overview

11.2 Plataforma Apache Spark:

Contiene los archivos referentes a la plataforma de Apache Spark debidamente instalados y configurados para su uso en este proyecto. Véase con calma los manuales (Usuario y técnico).

Enlace a repositorio Git: https://bitbucket.org/Yerfer/spark/overview

11.3 Plataforma Apache Hadoop:

Contiene los archivos referentes a la plataforma de Apache Hadoop debidamente instalados y configurados para su uso en este proyecto, específicamente el HDFS. Véase con calma los manuales (Usuario y técnico).

Enlace a repositorio Git: https://bitbucket.org/Yerfer/hadoop/overview

12 Referencias bibliográficas

[1] K. Shvachko, “HDFS Scalability: The limits to growth,” *Login*, pp. 6–16, 2010.

[2] D. Borthakur, “HDFS architecture guide,” *Hadoop Apache Proj.*, vol. 53, pp. 1–13, 2008.

[3] F. J. Salinas Dezerega, “EVALUACIÓN DE ALGORITMOS DE AGRUPAMIENTO UTILIZANDO APACHE SPARK,” 2016.

[4] C. Gómez Martinez, “Procesamiento de grandes volúmenes de datos en entornos Cloud Computing utilizando Hadoop MapReduce,” Universidad de Almería, 2013.

[5] M. Weihua, Z. Hong, L. Qianmu, and X. Bin, “Analysis of information management and scheduling technology in Hadoop,” vol. 12, no. 2, pp. 133–138, 2014.

[6] N. Rodríguez, M. Murazzo, D. Villafañe, M. Alves, and D. Mede, “Integración de Computación Heterogénea con Hadoop para Cloud Computing,” *XV Work. Investig. EN CIENCIAS LA Comput.*, pp. 1004–1008, 2013.

[7] M. A. Murazzo, N. R. Rodriguez, M. Guevara, and F. G. Tinetti, “Identificación de Algoritmos de Cómputo Intensivo para Big Data y su Implementación en Clouds,” 2016.

[8] R. Leira, P. Roquero, C. Vega, I. González, and J. Aracil, “HPSEngine: Motor de alto rendimiento y baja latencia para el procesamiento distribuido en tiempo real,” *research.cvega.es*.

[9] D. Ramírez Sánchez, “Data Mining sobre logs procesados con Hadoop,” BARCELONATECH, 2014.

[10] S. García, S. Ramírez-Gallego, J. Luengo, and F. Herrera, “Big Data : Preprocesamiento,” *novática*, pp. 17–23, 2016.

[11] S. J. Valbuena and J. M. Londoño, “SISTEMAS PARA ALMACENAR GRANDES VOLÚMENES DE DATOS,” *Rev. GTI*, vol. 13, no. 37, 2015.

[12] A. Hernández Dominguez and A. Hernández Yeja, “Acerca de la aplicación de MapReduce + Hadoop en el tratamiento de Big Data,” *Rev. Cuba. Ciencias Informáticas*, vol. 9, no. 3, pp. 49–62, 2015.

[13] S. Moon, J. Lee, X. Sun, and Y. suk Kee, “Optimizing the Hadoop MapReduce Framework with high-performance storage devices,” *J. Supercomput.*, vol. 71, no. 9, pp. 3525–3548, Sep. 2015.

[14] A. F. Rojas Hernandez and N. Y. Gelvez Garcia, “Distributed processing using cosine similarity for mapping Big Data in Hadoop,” *IEEE Lat. Am. Trans.*, vol. 14, no. 6, pp. 2857–2861, 2016.

[15] E. Ruiz García, “ESTUDIO Y EVALUACIÓN DE SISTEMAS ‘BIG DATA’ DE TRATAMIENTO DE INFORMACIÓN,” Universidad Politécnica de Madrid, 2016.
[16] E. F. Silva Balocchi, “Análisis y comparación entre el motor de bases de datos orientado a columnas Infobright y el framework de aplicaciones distribuidas Hadoop en escenarios de uso de bases de datos analíticas.” Universidad de Chile, 2014.
[17] M. Alcívar, E. Espinoza, and V. Cedeño, “Análisis de la información de una Base de Datos Transaccional usando Hive sobre Hadoop,” 2011.
[18] L. F. Tabares and J. F. Hernández, “Big Data Analytics : Oportunidades , Retos y Tendencias,” *Univ. San Buenaventura*, p. 20, 2014.
[19] R. Ramos Pollán, A. Cruz Roa, and F. A. González, “A Framework for High Performance Image Analysis Pipelines over Cloud Resources,” in *7th Colombian Computing Congress, CCC 2012 - Conference Proceedings*, 2012.
[20] M. Trigila and R. Di Pasquale, “ENSEÑANDO COMPUTACIÓN PARALELA CON CLUSTERES DE BAJO COSTO - TEACHING PARALLEL COMPUTING WITH LOW-COST CLUSTER,” *XIV Int. Conf. Eng. Technol. Educ.*, 2016.
[21] A. N. Cadavid, J. D. Fernández Martínez, and J. Morales Vélez, “Revisión de metodologías ágiles para el desarrolldo de software A review of agile methoologies for software development,” *Univ. Icesi*, vol. 11 No. 2, pp. 30–39, 2013.
[22] A. Firdaus, I. Ghani, and N. I. M. Yasin, “Developing Secure Websites Using Feature Driven Development (FDD): A Case Study,” *J. Clean Energy Technol.*, vol. 1, no. 4, pp. 322–326, 2013.
[23] A. Molpeceres, “Procesos de desarrollo: RUP, XP y FDD,” *javaHispano org*, 2003.
[24] F. Alvarez Goikoetxea, “Estudio de un entorno de computación distribuida con Hadoop,” 2016.
[25] T. Lee, H. Lee, K. H. Rhee, and S. U. Shin, “The efficient implementation of distributed indexing with hadoop for digital investigations on big data,” *Comput. Sci. Inf. Syst.*, vol. 11, no. 3, pp. 1037–1054, 2014.
[26] A. Plasencia Salgueiro and B. de los M. Ballagas Flores, “Análisis comparativo de herramientas de recuperación y análisis de información de acceso libre desde una concepción docente,” *Transinformação*, vol. 26, no. 3, pp. 315–326, Dec. 2014.
[27] V. I. Rosa Urrutia and J. G. Rivera Pleitez, “Big Data , análisis de datos en la nube,” *Entorno*, pp. 17–24, 2016.
[28] J. J. Camargo Vega, J. F. Camargo Ortega, and L. Joyanes Aguilar, “Conociendo Big Data,” *Rev. Fac. Ing. (Fac. Ing.), Enero-Abril*, vol. 24, no. 38, pp. 63–77, 2015.
[29] M. Niño and A. Illarramendi, “ENTENDIENDO EL BIG DATA: ANTECEDENTES, ORIGEN Y DESARROLLO POSTERIOR,” *DYNA NEW Technol.*, vol. 2, no. 3, pp. 1–8, 2015.
[30] D. Laney, “3D Data Management: Controlling Data Volumen, Velocity, and Variety,” *Appl. Deliv. Strateg.*, vol. 949, no. February 2001, p. 4, 2001.
[31] A. Ghaffar Shoro and T. Rahim Soomro, “Big Data Analysis: Ap Spark Perspective,” *Glob. J. Comput. Sci. Technol.*, vol. 15, no. 1, pp. 7–14, 2015.
[32] A. Martín, S. Chavez, N. Rodriguez, A. Valenzuela, and M. Murazzo, “Bases de datos NoSql en cloud computing,” *XV Work. Investig. EN CIENCIAS LA Comput.*, pp. 166–170, 2013.
[33] M. N. Vora, “Hadoop-HBase for large-scale data,” *Proc. 2011 Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2011*, vol. 1, pp. 601–605, 2011.
[34] C. M. Pérez Ibarra, S. A. Mendez, and N. M. Pérez Otero, “Restauración de imágenes y metaheurísticas en Hadoop,” 2015.
[35] O. O’Malley, “Terabyte sort on apache hadoop,” *Yahoo*, pp. 1–3, 2008.
[36] J. Maillo, I. Triguero, and F. Herrera, “Un enfoque MapReduce del algoritmo k-vecinos más cercanos para Big Data,” *ACM*, 2003.

[37] E. Izaguirre Equiza, “Trabajo Fin de Grado: Sistemas de recomendación en Apache Spark,” 2015.
[38] J. M. García Sánchez, “Sparkanalyzer: Instrumentación de Apache Spark,” 2016.
[39] A. Romero Rodríguez, “Extracción de Información Semántica a Partir de Categorías de Texto Estado del Arte.”
[40] L. A. López Rodríguez, J. M. Quinteiro González, and P. Hernández Morera, “Clasificador basado en Análisis de Sentimiento.”
[41] T. Deselaers, L. Pimenidis, and H. Ney, “Bag-of-visual-words models for adult image classification and filtering,” *2008 19th Int. Conf. Pattern Recognit.*, pp. 1–4, 2008.
[42] Y. Zhang, R. Jin, and Z. H. Zhou, “Understanding bag-of-words model: A statistical framework,” *Int. J. Mach. Learn. Cybern.*, vol. 1, no. 1–4, pp. 43–52, 2010.
[43] L. Wu, S. C. H. Hoi, and N. Yu, “Semantics-preserving bag-of-words models and applications,” *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1908–1920, 2010.
[44] C. A. Cobos Lozada, E. E. Mendoza, M. E. Mendoza Becerra, L. C. Gómez Flórez, and E. L. Guzmán, “Algoritmos de expansión de consulta basados en una nueva función discreta de relevancia. (Spanish),” *UIS Ing.*, vol. 10, no. 1, pp. 7–20, 2011.
[45] S. Jaramillo Valbuena and J. M. Londoño, *BÚSQUEDA DE DOCUMENTOS BASADA EN EL USO DE ÍNDICES ONTOLÓGICOS CREADOS CON MAPREDUCE*, vol. 24, no. 1. Bogotá: Universidad Militar, Nueva Granada, 2014.
[46] S. Vázquez-Reyes, M. de León-Sigg, P. Velasco-Elizondo, J. Villa-Cisneros, and S. Briceño-Muro, “Recuperación de información a través del índice invertido en Be Intelligent,” *RISTI - Rev. Iber. Sist. e Tecnol. Inf.*, no. 21, pp. 85–98, 2017.
[47] S. D. Amaro Calderón and J. C. Valverde Rebaza, “Metodologías Ágiles,” *Esc. Informatica.*, pp. 1–37, 2007.
[48] K. Mendes Calo, E. C. Estevez, and P. R. Fillottrani, “Evaluación de metodologías ágiles para desarrollo de software,” *XII Workshop de Investigadores en Ciencias de la Computación*. 2010.
[49] D. D. Lewis, “Reuters-21578 text categorization test collection.” AT&T Labs - Research, 2004.
[50] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning Word Vectors for Sentiment Analysis,” *Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol.*, pp. 142–150, 2011.
[51] K. Lang, “NewsWeeder: Learning to Filter Netnews,” *Proc. 12th Int. Mach. Learn. Conf.*, pp. 331–339, 1995.
[52] F. Galgani and A. Hoffmann, “LEXA: Towards Automatic Legal Citation Classification,” in *AI 2010: Advances in Artificial Intelligence*, 2010, vol. 6464, pp. 445–454.
[53] F. Galgani, P. Compton, and A. Hoffmann, “Towards automatic generation of catchphrases for legal case reports,” in *the 13th International Conference on Intelligent Text Processing and Computational Linguistics*, 2012, vol. 7182, pp. 415–426.
[54] F. Galgani, P. Compton, and A. Hoffmann, “Knowledge Acquisition for Categorization of Legal Case Reports,” in *PKAW 2012*, 2012, vol. LNAI 7457, pp. 118–132.
[55] F. Galgani, P. Compton, and A. Hoffmann, “Citation Based Summarisation of Legal Texts,” in *PRICAI 2012*, 2012, vol. LNCS 7458, pp. 40–52.
[56] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques,” in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.