UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

ESCUELA DE POSTGRADO

MAESTRÍA DE CIENCIAS MENCIÓN INFORMÁTICA

TESIS

"ARQUITECTURA HOMOGÉNEA Y HETEROGÉNEA PARA EL PROCESAMIENTO DISTRIBUIDO DE DATOS NO ESTRUCTURADOS CON FRAMEWORK HADOOP"

Para Optar el Grado Académico de Maestro en Ciencias mención Informática

Presentado por: BR. Claudio Isaias Huancahuire Bravo

Asesor: Mg. Javier Arturo Rozas Huacho.

Cusco – Perú

2021

DEDICATORIA

Dedico a Yimi Isaias y Lian Albertina por ser mis razones más intensas de superación, entre disgustos y alegrías son fuente diaria de mi existencia. Como también a Lucio Marciano Huancahuire Quisca- Físico-Matemático de la Universidad Católica de Santa María (UCSM), que me indico un parámetro “mi herencia”, es tu profesión.

AGRADECIMIENTOS

Gracias a mis abuelos Isaias Bravo Valenzuela y Albertina Mendoza Ramírez que sacrificaron sus costumbres, tradiciones y sueños de su tierra Antabamba. Por la atención, servicio y superación de nosotros sus nietos.

A mi hermano Guido que su sugerencia ser un Catedrático “Es la máxima expresión profesional”.

A mi hermano Guildo, que cree en mí hasta el último mínimo.

A mi hermana Norka, con su atención infaltable de su servicio, positivismo y confianza.

Índice general

Lista de tablas

Lista de figuras

Resumen

Las limitaciones digitales de almacenamiento estructurado de filas y columnas de Microsoft-Excel, fueron resueltas por sistemas denominados Administradores de Sistema de Base de Datos Relacional-RDBMS, que integra modelo Entidad-Relación y el Lenguaje Estructurado de Consultas-SQL de Microsoft SQL-Server, MySql, PostgresSQL y Oracle SQL Developer. Estos datos estructurados fueron mejorados con ETL (extraer, transformar y cargar) en Data Warehouse (almacén de datos) todo ello enmarcado en arquitectura y software convencional y tradicional. Ahora interactuamos con Big Data con la limitación de exponencialmente grandes datos, por rendimiento y precios exorbitantes para su tratamiento.

La limitación de almacenamiento tradicional y convencional es Big Data. “En el año 2011 se pretendía cuantificar la cantidad de información generada y almacenada en el mundo. Google, con Eric Schmidt, afirmó que la Humanidad había creado hasta 2003 una cantidad equivalente a 5 Exabytes, añadiendo que ahora esta cifra se generaba en 2 días”. (https://www.sciencemag.org/). Las diferentes fuentes como: La web 2.0 que dio el proceso de interactuar con el almacenamiento de datos en línea de transacciones, ventas, compras, los sensores, IoT (Internet de las Cosas), es ineludible contar con las redes sociales digitales y además el avance del TIC de un teléfono fijo a un "Smartphone". Datos de **diferentes tipos** como son: Estructurado, semi-estructurado y no estructurado que son: Correos electrónicos, log, un documento de algún procesador de palabra, hojas electrónicas, una imagen, un objeto, blogs, mensajes de correo de voz, mensajes instantáneos, contenidos Web, audios y videos.

Todo esto enmarcado en Big Data a precios no asequibles para los diferentes sectores, El desafío de Big Data es la captura, almacenamiento y tratamiento estratégico de esa exorbitante información a un precio módico para una óptima toma de decisión en tiempo real.

La propuesta es diseñar e implementar una arquitectura homogénea y heterogénea basado en el framework Hadoop, bajo el modelo cliente/servidor en base a Hardware Commodity, generando así clúster homogénea y heterogénea. Dichos clústeres son tolerantes a fallos, acoplado su parte lógica con el modelo de programación MapReduce y luego almacenar los datos no estructurados en sistema de archivos distribuidos HDFS ubicados en nodos esclavos y dichos nodos al ser adicionados con memoria RAM, disco duro y E/S entrada/salida de datos hace que los tiempos de rendimiento disminuya tanto en clúster homogénea y heterogénea que permite procesar grandes volúmenes de datos con la función Map que convierte en números de pares clave/valor y la función Reduce procesa las tuplas clave/valor que llegan de la función Map los reduce para su salida final. La mejora de esta arquitectura con Framework Apache Hadoop es YARN (otro Administrador de recursos que mejora la versión anterior) por parte de Hadoop versión 2, que coordina exclusivamente los recursos del clúster y administra que bloques se distribuyen en discos locales los nodos esclavos y por último el almacenamiento en HDFS (Sistema de Archivos Distribuidos sobre Hadoop) que es donde se almacena los bloques y réplicas en cada nodo esclavo y el aprovisionamiento, administración y monitoreo de todo la arquitectura en conjunto como uno solo servidor-clúster. Logrando la característica escalable horizontalmente en arquitectura homogénea y heterogénea.

Palabras claves: Big data, MapReduce, HDFS, semiestructurado, no estructurado, YARN, Hadoop, Apache, nodos

Abstract

The digital limitations of structured storage of rows and columns of Microsoft-Excel, were solved by systems called System Administrators of Relational Database System-RDBMS, which integrates the Entity-Relationship model and the Structured Query Language-SQL of Microsoft SQL- Server, MySql, PostgresSQL and Oracle SQL Developer. This structured data was improved with ETL (extract, transform and load) in the Data Warehouse, all framed in conventional and traditional architecture and software. Now we interact with Big Data with the limitation of capturing, analyzing semi-structured and exponentially large unstructured data, for performance and exorbitant prices for processing.

The limitation of traditional and conventional storage is Big Data. “In 2011 it was intended to quantify the amount of information generated and stored in the world. Google, with Eric Schmidt, said that Humanity had created an amount equivalent to 5 Exabytes until 2003, adding that this figure was now generated in 2 days.” (https://www.sciencemag.org/). The different sources such as: Web 2.0 that gave power to interact with the online data storage of transactions, sales, purchases, sensors, IoT (Internet of Things), it is unavoidable to have digital social networks and also the advance of the TIC from a landline to a "Smartphone". Data of different types such as: Structured, semi-structured and unstructured which are: Emails, text files, log, a document of some word processor, electronic sheets, a image, an object, blogs, voicemail messages, instant messages, Web content, audios and videos.

All this framed in Big Data at prices not affordable for different sectors.

The challenge of Big Data is the capture, storage and strategic treatment of that exorbitant information at a reasonable price for optimal decision making in real time. The proposal is design and implement a homogeneous and heterogeneous architecture based on the Hadoop framework, under the client / server model based on Hardware Commodity, thus

generating a homogeneous and heterogeneous cluster. These clusters are fault tolerant, their logical part is coupled with the MapReduce programming model and then storing the unstructured data in the HDFS distributed file system located in slave nodes and said nodes when added with RAM memory, hard disk and input and output of data causes performance times to decrease both in homogeneous and heterogeneous cluster that allows to process large volumes of data with the Map function that converts into numbers of key / value pairs and the Reduce function processes the key / value tuples arriving from the Map function reduces them for final output. The improvement of this architecture with Framework Apache Hadoop is YARN (another Resource Manager which improves the previous version) by Hadoop version 2, which coordinates exclusively the resources of the cluster and manages which blocks are distributed on local disks of the slave nodes and by Ultimately the storage in HDFS (Distributed File System on Hadoop) which is where the blocks and copies of these blocks are stored the amount of in each slave node and the provisioning, administration and monitoring of the entire architecture as a single server-cluster.

Keywords: Big data, MapReduce, HDFS, semi-structured, unstructured, YARN, Hadoop, Apache, nodes.

INTRODUCCIÓN

Debido al avance tecnológico de almacenamiento de datos digitales en el mundo actual se incrementan de manera exorbitante dichos datos englobando en Big Data, generando un nuevo desafío para almacenar, procesar y analizar un gran volumen de datos. Las tecnologías tradicionales no se convierten en una solución adecuada para procesar. Datos e información irrefutablemente esenciales en esta sociedad digital para tomar decisiones en contexto de Industria, Salud, Transporte, Universidades, Empresas, MYPES y Negocios.

Los datos se almacenaban únicamente con formatos estructurados en Excel, Spss, SQL-Server, Oracle, MySql, etc. con equivalencia al 20%. Sin embargo, desde los formatos semi-estructurados y no estructurados. Provenientes de Internet, Web 2.0, Google, Amazon, Redes Sociales (Facebook, Twitter, WhatsApp) técnicamente definidos como PetaByte, ExaByte, ZettaByte, YottaByte, BrontoByte los trata exclusivamente Big Data. Son equivalentes al 80%. Por ello es inevitable e innegable la proliferación exponencialmente alta, de naturaleza heterogénea y con la velocidad a la que se generan dificultan una toma de decisión óptima. La solución diseñada no es por medio de una supercomputadora de Empresas como IBM, Amazon Web Service, Oracle, Microsoft Azuze, Cloudera y HortonWorks. Sino a través de computadoras de precios módicos unidas en redes utilizando tecnología Hadoop (código abierto) la cual permite procesar a las aplicaciones en miles de nodos escalables horizontalmente.

Apache Hadoop se ha convertido en la plataforma elegida para el desarrollo de aplicaciones de datos a gran escala. Utilizando desde uno nodo a 4.000 nodos en la versión 1 de Hadoop. Ahora mejora su rendimiento con la administración de recurso YARN.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. Situación problemática.

Todos los días, generamos 2,5 trillones de bytes, data “en el año 2011, toda la información del mundo se ha generado hasta el año 2003, es equivalente a 5 Exabyte, añadiendo que ahora esa cifra se genera en dos días”, este crecimiento exponencial se estima que aumentarán de 2.8 ZB en 2012 a 40 ZB para 2020 a gran escala. (Aguilar, 2013, pág. 15).

Cada día estos datos exponenciales proceden de diferentes fuentes y sitios digitales como son: Sensores utilizados para recoger información del clima, entradas (posts) en sitios de medios sociales (YouTube, Facebook, Twitter, WhatsApp y Google+). videoconferencias a través de Skype, fotografías (Instagram) y videos y series en streaming (NetFlix), registros de transacciones comerciales en la Web 2.0 y señales GPS de teléfonos celulares, por citar unas cuantas fuentes de datos extremadamente exponenciales.

A medida que los datos crecen en grandes volúmenes de datos, únicamente compañías tecnológicas propietarias pueden hacer uso de este tratamiento de Big Data. Existen muchas Empresas y Organizaciones que no pueden pagar y tener una arquitectura con licencia libre para estos datos.

Indefectiblemente no es problema únicamente de grandes compañías como Google, Amazon, Facebook, IBM, Oracle y Microsoft. Que cuentan con Clúster HA- clúster de alta disponibilidad) La gestión de grandes cantidades de datos semiestructurados y no estructurados enmarcado todo ello dentro de Big Data, ahora está en todos los sectores: como la Agricultura, Industria y Construcción, Educación, Transporte, Turismo, Económica, Salud y en Organizaciones y Empresas. y no es posible administrar la ingente

cantidad de datos por tecnología y sistema de gestión de base de datos convencionales y tradicionales, creando la necesidad de obtener permanentemente un supercomputador o servidor de altas prestaciones y escalar vertical por precio exorbitante.

1.2. Formulación del problema.

1.2.1. Problema general.

Aumentando el número NODOS EN CLUSTER de arquitecturas homogéneas y heterogéneas para el procesamiento de datos no estructurados sobre Hadoop Distributed File System-HDFS, mejorara el TIEMPO DE RENDIMIENTO EN CADA CLUSTER, al ejecutar con MapReduce.

1.2.1. Problemas específicos

- ¿Se cuenta con un diseño de almacenamiento y procesamiento distribuido de datos no estructurados con Hadoop Distributed File System (HDFS), para estimar tiempos de procesamiento distribuido de grandes volúmenes de datos no estructurados, en arquitectura homogéneas y heterogéneas?
- ¿Existe una correlación de funciones matemáticas que permitan estimar el tiempo de rendimiento en el procesamiento distribuido de datos no estructurados de grandes volúmenes de datos, mediante la implementación de dos arquitecturas de clúster homogénea y heterogénea, usando MapReduce?

1.3. Justificación de la Investigación

El advenimiento de la sociedad del conocimiento y el impacto de las emergentes tecnologías de la información, tal como: internet de las cosas, el uso creciente de servicios de “Cloud computing”, interacción organizacional creciente a través de internet, etc.;

Generan grandes volúmenes de información, dando lugar al concepto “BIG DATA”. El desafío que enfrentan las instituciones, es procesar estos exorbitantes volúmenes datos, a partir del cual obtener ventajas competitivas.

En este contexto una de las primeras características de estos exorbitantes volúmenes de datos, es que están constituidos mayoritariamente por datos no estructurados, que requiere de tecnologías que implican innovaciones que van más allá de los procesos convencionales, como almacenamiento convencional (datos estructurados) con operaciones tradicionales, soportados en servidores de base de datos de alta disponibilidad y de escalabilidad vertical, de costos muy elevados.

En consecuencia, se hace imperativo, proponer alternativas que optimicen o innoven las tecnologías convencionales. En este sentido, una alternativa importante es utilizar tecnologías de clúster de computadoras, que permitan agrupar hardware básico que este distribuido, sea tolerante a fallos, escalable horizontalmente y, de bajo costo.

Sobre esta tecnología, las instituciones pueden estimar la mejor configuración para reducir tiempos de procesamiento, variando número de nodos, ya sea en una arquitectura homogénea o heterogénea; que les posibilite implementar soluciones que les permita procesar grandes volúmenes de datos estructurados y no estructurados, y mejorar así el proceso de toma de decisiones, que les permita interactuar con algunas ventajas competitivas en este entorno altamente globalizado.

Es en este panorama que radica la importancia de este trabajo, porque plantea los delineamientos que les permitirá a las organizaciones, estimar a priori los tiempos de procesamiento distribuido de grandes volúmenes de datos no estructurados con framework Hadoop, en arquitecturas homogéneas y heterogéneas; que coadyuvará a la mejor toma de decisiones.

1.4. Objetivos de la Investigación

1.4.1. Objetivo General.

Mejorar la correlación de los tiempos de rendimiento en clúster de arquitecturas homogéneas y heterogéneas para el procesamiento de datos no estructurados sobre Hadoop Distributed File System-HDFS, para diferentes números de nodos; mediante la implementación con MapReduce

1.4.2. Objetivo Específico.

- Diseñar la estructura de almacenamiento y procesamiento distribuido de datos no estructurados con Hadoop Distributed File System (HDFS), en arquitecturas homogéneas y heterogéneas; y ejecutar el proceso paralelo con modelo de programación MapReduce.
- Determinar la correlación de funciones matemáticas que permitan estimar el tiempo de rendimiento en el procesamiento distribuido de datos no estructurados de grandes volúmenes de datos, mediante la implementación de dos arquitecturas de clúster homogénea y heterogénea, usando MapReduce.

CAPÍTULO II

MARCO TEÓRICO CONCEPTUAL

2.1 Bases Teóricas

2.1.1. Unidades de medida en informática

Las unidades equivalentes como el kilobyte (KB), megabyte (MB) y gigabyte (GB) y terabyte (TB) se utilizan comúnmente para expresar el tamaño de almacenamiento convencional. La Tabla 1 compara los valores, nombres y símbolos.

Tabla 1 Unidades de medida en informática.

Equivalencia	***Medida***	***Simbología***
1024 Byte	1 KiloByte	KB
1024 KB	1 MegaByte	MB
1024 MB	1 GigaByte	GB
1024 GB	1 TeraByte	TB
1024 TB	1 PetaByte	PB
1024 PB	1 ExaByte	EB
1024 EB	1 ZettaByte	ZB
1024 ZB	1 YottaByte	YB
1024 YB	1 BrontoByte	BB

Nota: Unidades decimales de medida para Big Data. (https://www.ibm.com).

Actualmente en 60 segundos, almacena referencia figura 1. (http://bit.ly/smartlibrary)

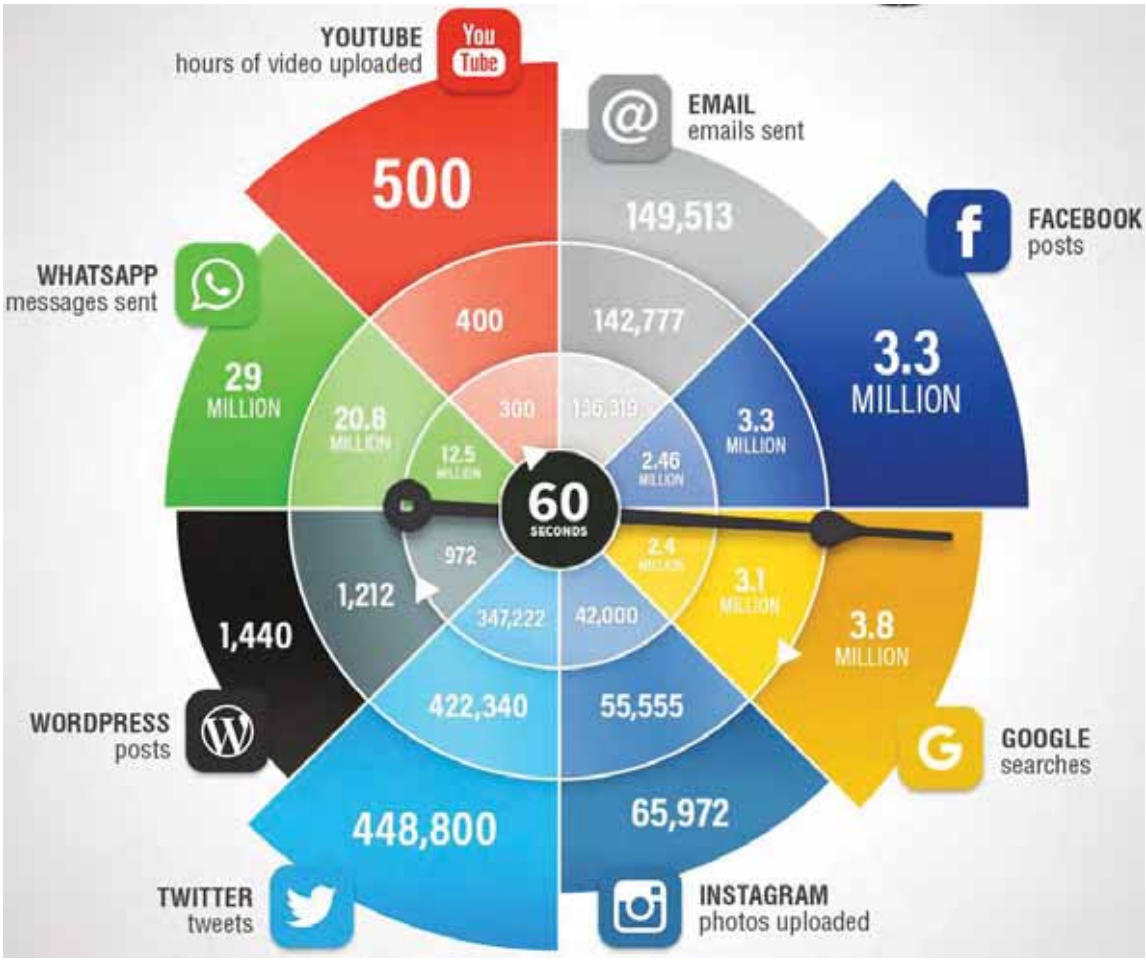


Figura 1. Datos en 60 segundos (http://bit.ly/smartlibrary)

2.1.2. Tipo de Datos

- DATOS ESTRUCTURADOS

  La mayoría de fuentes de datos tradicionales son datos estructurados, datos con formato o esquema fijo que poseen campos fijos, que se especifica en detalle. Formatos típicos son: fecha de nacimiento, DNI, cuenta corriente en un banco y etcétera. (Aguilar, 2013, pág. 28).

- DATOS SEMIESTRUCTURADOS

  Los datos semiestructurados tienen un flujo lógico y un formato que puede ser definido, pero no es fácil su comprensión por el usuario. Datos que no tienen

formatos fijos, pero contienen etiquetas y otros marcadores que permiten separar los elementos dato. Un ejemplo típico son los registros Web logs. Un Web log se compone de diferentes piezas de información, cada una de las cuales sirve para un propósito específico. Ejemplos típicos son el texto de etiquetas de lenguajes XML y HTML. (Aguilar, 2013)

- DATOS NO ESTRUCTURADOS

  Los datos no estructurados son datos sin tipos predefinidos. Se almacenan como "documentos" u "objetos" sin estructura uniforme, y se tiene poco o ningún control sobre ellos. Por ejemplo, las imágenes se clasifican por su resolución en píxeles. Datos que no tienen campos fijos; ejemplos típicos son: audio, video, fotografías, documentos impresos, cartas, hojas electrónicas, imágenes digitales, formularios especiales, mensajes de correo electrónico y de texto, formatos de texto libre como correos electrónicos, mensajes instantáneos SMS, artículos, libros, mensajes de mensajería instantánea tipo WhatsApp, Line, Joyn, Viber, Line, WeChat, Spotbros. Al menos, el 80% de la información de las organizaciones no reside en las bases de datos relacionales o archivos de datos, sino que se encuentran esparcidos a lo largo y ancho de la organización. (Aguilar, 2013, pág. 29)

2.1.3. Definición Big Data.

Big data es un término de origen inglés cuya traducción equivale a "Datos masivos", aun cuando no existe una definición formal para Big Data, líderes en el mercado de TI lo exponen como (Mai, 2016, pág. 8), como referencia figura 2.

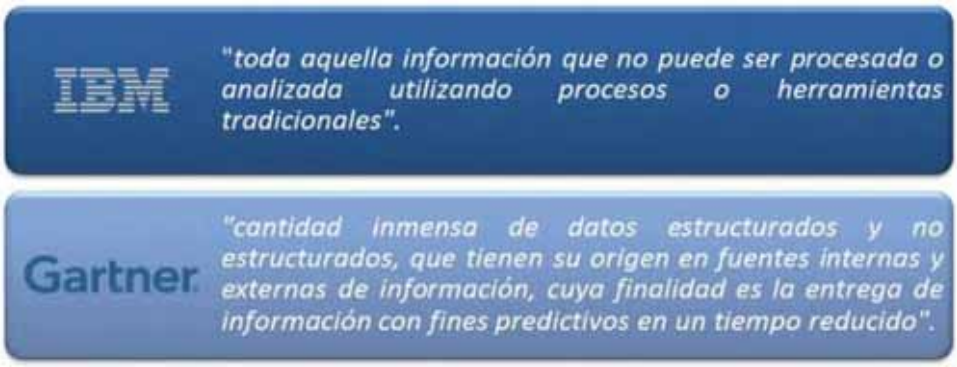


Figura 2. Definición de IBM y Gartner (Mai, 2016)

2.1.4. Las dimensiones de Big Data.

Primero se definieron las tres dimensiones de Big Data –'las tres V:' volumen, variedad y velocidad. Estas tres dimensiones engloban los principales atributos de Big Data, como referencia figura 3.

Figura 3. Las 3 V's principales de Big Data (http://datacrunchers.eu/)

Luego se desarrolló una importante dimensión más: la veracidad. (Mai, 2016, pág. 8) La convergencia de estas cuatro dimensiones ayuda tanto a definir como a distinguir Big Data (Mai, 2016):

- Volumen: El tamaño de información. Es la característica que se asocia con mayor frecuencia a Big Data, el volumen hace referencia a las cantidades masivas de datos que las organizaciones intentan aprovechar para mejorar la

toma de decisiones en toda la empresa. Los volúmenes de datos continúan aumentado a un ritmo sin precedentes. Algo más de la mitad de los encuestados consideran que conjuntos de datos de entre un terabyte y un petabyte ya son big data. Aun así, todos ellos estaban de acuerdo en que sea lo que fuere que se considere un "volumen alto" hoy en día, mañana lo será más. (Mai, 2016, pág. 9). Como referencia figura 4.

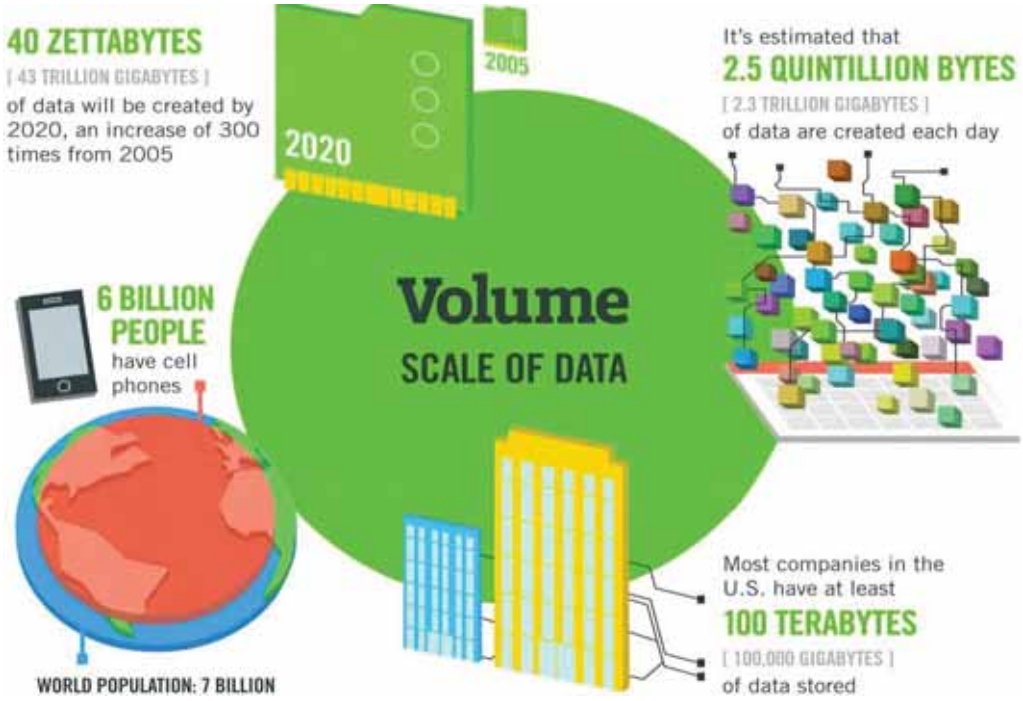


Figura 4. El volumen de datos. (https://www.ibm.com)

- Velocidad: Incluye tanto la medida de velocidad en la que llegan los datos y también el tiempo en que se debe actuar. La velocidad a la que se crea, procesa y analiza los datos continúa aumentando. Contribuir a una mayor velocidad es la naturaleza en tiempo real de la creación de datos, así como la necesidad de incorporar datos en streaming a los procesos de negocio y la toma de decisiones. La velocidad afecta a la latencia: el tiempo de espera entre el momento en el que se crean los datos, el momento en el que se captan y el momento en el que están accesibles. Hoy en día, los datos se generan de forma continua a una velocidad

a la que a los sistemas tradicionales les resulta imposible captarlos, almacenarlos y analizarlos. Para los procesos en que el tiempo resulta fundamental, como la detección de fraude en tiempo real o el marketing "instantáneo", ciertos tipos de datos deben analizarse en tiempo real así resulten útiles para el negocio, figura 5. (Mai, 2016, pág. 9).

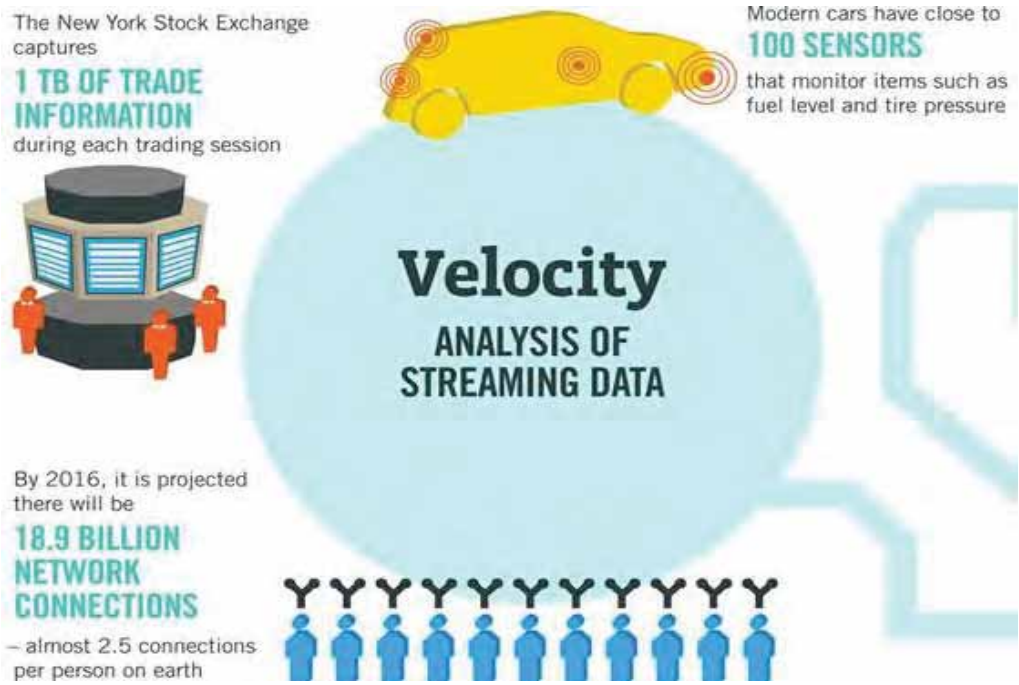


Figura 5. La velocidad de datos. (https://www.ibm.com)

- Variedad: Se refiere a la heterogeneidad de los datos, tiene que ver con gestionar la complejidad de múltiples tipos de datos, incluidos los datos estructurados, semiestructurados y no estructurados. Las organizaciones necesitan integrar y analizar datos de un complejo abanico de fuentes de información tanto tradicional como no tradicional procedentes tanto de dentro como de fuera de la empresa. Con la profusión de sensores, dispositivos inteligentes y tecnologías de colaboración social, los datos que se generan presentan innumerables formas entre las que se incluyen texto, datos web, tuits, datos de sensores, audio, vídeo,

secuencias de clic, archivos de registro y mucho más, figura 6. (Mai, 2016, pág. 9).

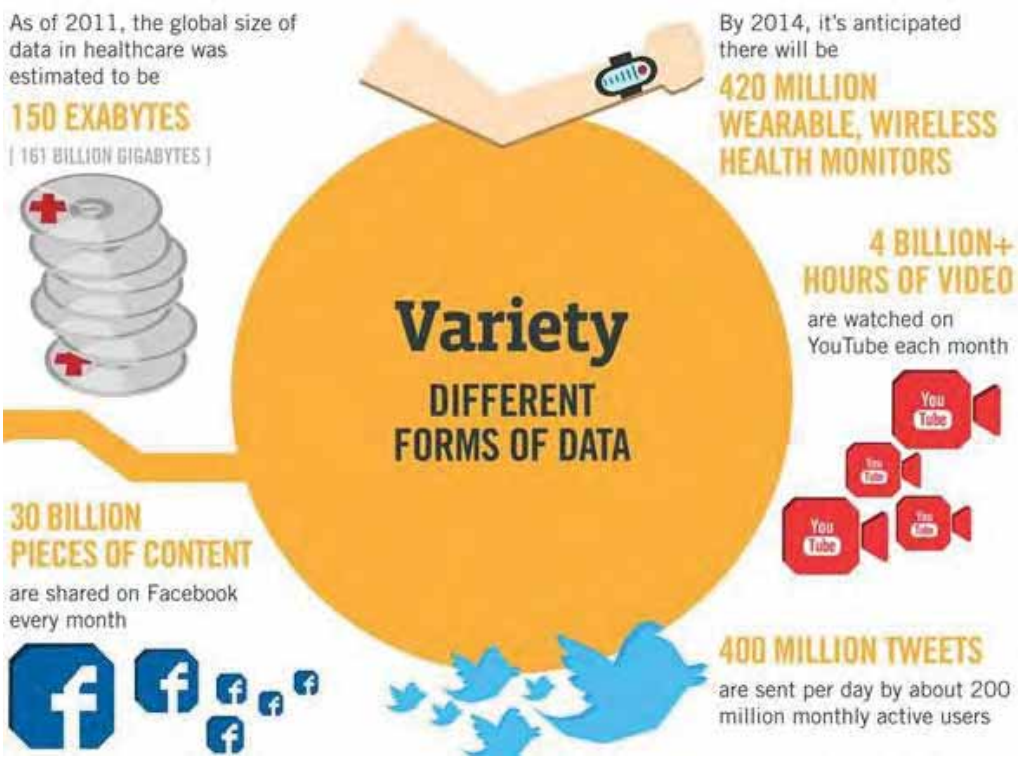


Figura 6. La variedad de datos. (https://www.ibm.com)

- Veracidad: La veracidad hace referencia al nivel de fiabilidad asociado a ciertos tipos de datos. Esforzarse por conseguir unos datos de alta calidad es un requisito importante y un reto fundamental de Big Data, pero incluso los mejores métodos de limpieza de datos no pueden eliminar la imprevisibilidad inherente de algunos datos, como el tiempo, la economía o las futuras decisiones de compra de un cliente. La necesidad de reconocer y planificar la incertidumbre es una dimensión de Big Data que surge a medida que los directivos intentan comprender mejor el mundo incierto que les rodea. (Mai, 2016, pág. 9).

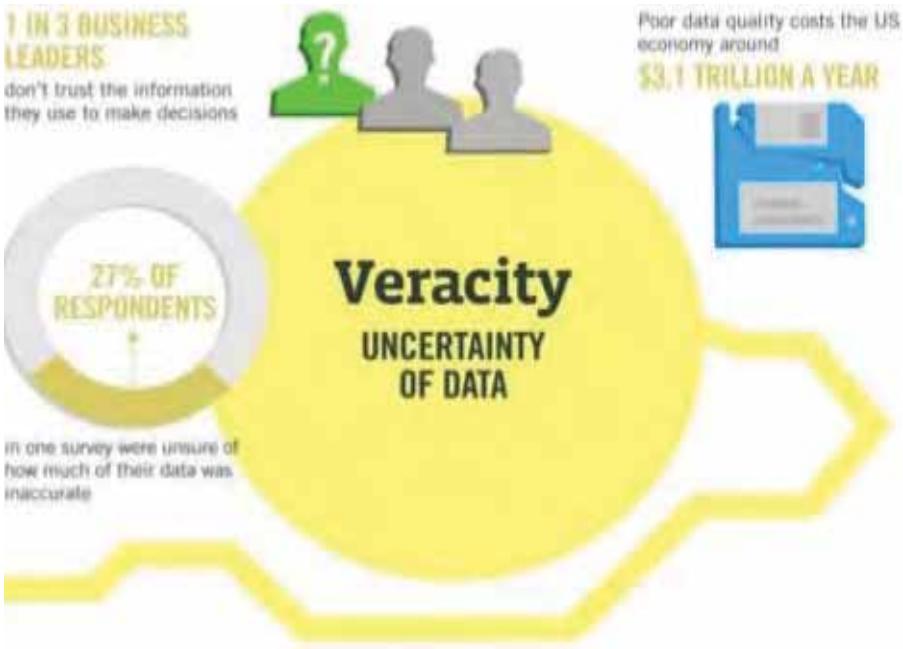


Figura 7. La veracidad de datos. (https://www.ibm.com)

2.1.5. Arquitectura de información de Big Data de Oracle

La arquitectura de información de Big Data consta de cuatro etapas: adquirir, organizar, analizar y decidir. En todo el proceso, se deberán tener presente las capas de gestión, seguridad y gobierno, referencia figura 8.

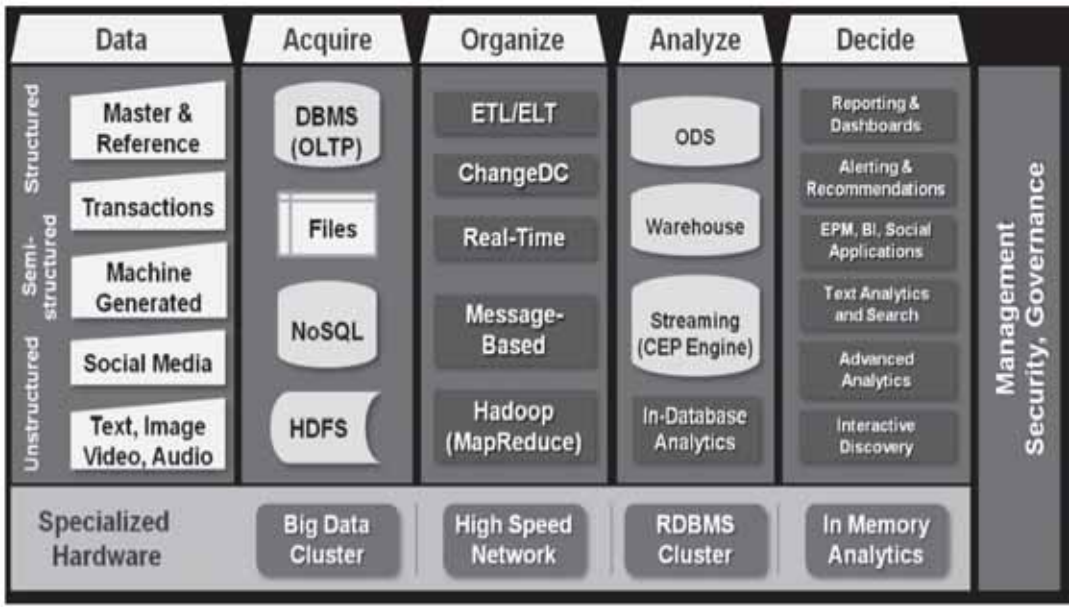


Figura 8. Arquitectura de Big Data de Oracle. (www.oracle.com)

2.1.6. Apache

"La Fundación Software Apache. Proporciona soporte para los proyectos de software de la Comunidad de código abierto". (http://hadoop.apache.org/).

2.1.7. El origen de Hadoop

Veamos con un ejemplo simple, en cuanto al tiempo y velocidad. Supongamos que nuestra computadora tiene un disco duro con una velocidad de acceso de 5000 Mbps (megabits por segundo) y capacidad de almacenamiento de 1 TB (terabyte). La lectura de 1 TB (1.000.000 MB) llevaría del orden de 200 segundos (1.000.000 MB/5.000 = 200 segundos). Supongamos ahora que, en lugar de un único disco duro, la información estuviera almacenada en 100 discos duros (10 GB/disco) conectados en paralelo, la lectura de 1 TB de información llevaría un tiempo de 2 segundos. Es decir, hubiésemos reducido drásticamente la velocidad de acceso.

¿Cómo se llegó a Hadoop? Google trabajaba desde primeros del siglo XXI en nuevos métodos para el acceso a la información, y sus trabajos se dirigían al tratamiento masivo de grandes volúmenes de datos, y en sistemas paralelo, tal y como hemos citado en el ejemplo anterior. (Aguilar, 2013, pág. 235).

2.1.8. Ecosistema Hadoop.

Hadoop tiene un ecosistema muy diverso, que crece día tras día, por lo que es difícil saber de todos los proyectos que interactúan con Hadoop, como referencia figura 9. (http://www.apache.org/).

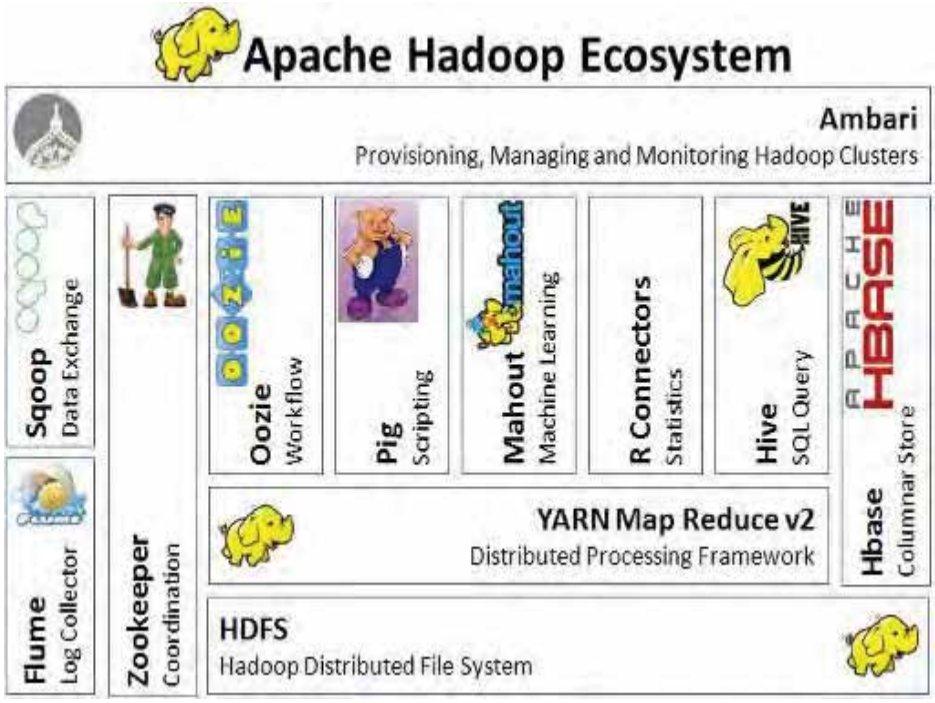


Figura 9. Herramientas del ecosistema Hadoop. (http://www.apache.org/)

2.1.9. Hadoop

Es un Framework que permite procesar en distribución y paralelo en un clúster sobre computadoras convencionales o estaciones de trabajo, lógicamente con MapReduce. Diseñado para escalar horizontalmente y es tolerante a fallas. (https://hadoop.apache.org/).

2.1.10. Módulos Hadoop.

- Hadoop Common: Son utilidades que admiten otros módulos. (Alapati, 2017, pág. 15)
- Sistema de archivos distribuidos de Hadoop (HDFS): Proporciona acceso de alto rendimiento a datos. (Alapati, 2017, pág. 15)
- YARN: Framework que planifica tareas y gestiona recursos. (Alapati, 2017, pág. 15)
- Hadoop MapReduce: Un sistema basado en YARN para el procesamiento paralelo de grandes conjuntos de datos. (Alapati, 2017, pág. 15)

2.1.11. Demonios de Hadoop versión 1.

a. NameNode (master): Es el nodo encargado de cierre, inicio y renombrado de directorios y ficheros. Es también el encargado de gestionar los metadatos y de la gestión y coordinación de bloques de datos. Posee el control y la información sobre la asignación de los bloques en los DataNodes y supervisión del número de réplicas existentes y su estado. Si hay pérdida de una réplica será el encargado de crear nueva replica en otro DataNode. (Alapati, 2017, pág. 16)
b. DataNode: Contiene bloques de información. Son los encargados de los procesos de lectura/escritura. En un sistema de archivos representados por datos y metadatos. DataNode identificados por ID de almacenamiento que interrelacionan a un NameNode para permitir su registro y acceso. Este sistema soporta 4000 DataNode máximo. (Alapati, 2017, pág. 16)
c. JobTracker: Coordina todo el trabajo e implementado en Java y su clase principal es JobTracker, existe un único JobTracker por clúster encargado de recibir todas las peticiones de clientes y organizar el trabajo para los TaskTrackers. Todos los TaskTracker deben enviar un paquete de control para tener informado al JobTracker. (Noelia, 2014, pág. 30)
d. TaskTracker: Encargado de realizar las tareas en las que divide el trabajo mediante la creación de distintos hilos que trabajen en paralelo. Estos pueden estar en tres estados no simultáneos, en reposo, trabajando o terminado. Su implementación en el caso de Hadoop también es Java y su principal clase es TaskTracker, referencia figura 10. (Noelia, 2014, pág. 40)

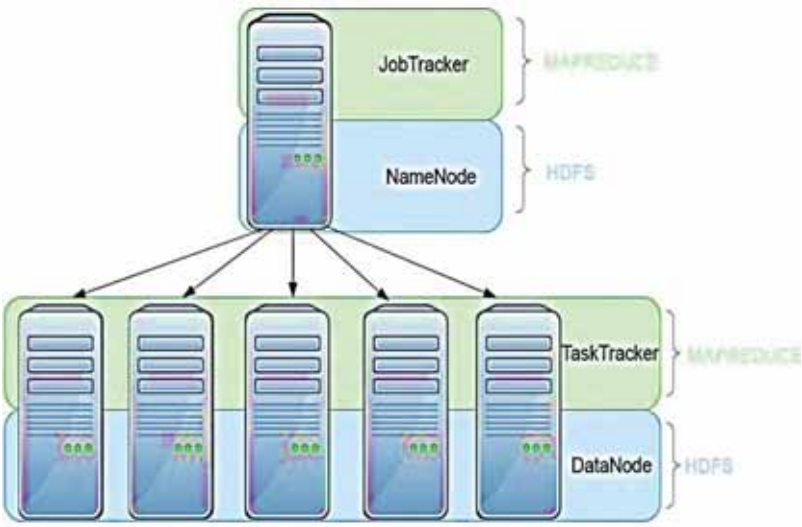


Figura 10. Arquitectura HDFS y MapReduce (Noelia, 2014)

2.1.12. Tipos de instalación de Hadoop

a. Independiente.

   Los servicios de Hadoop se ejecutan en una única JVM, Hadoop usa el sistema de archivos local y no HDFS para almacenar sus datos, y los trabajos de MapReduce se ejecutan con un solo asignador y un solo reductor. (https://hadoop.apache.org/)

b. Pseudo-distribuido.

   Esta es una simulación de un clúster multinodo real. Configura todos los servicios como en el caso de un clúster totalmente distribuido, pero todos los demonios (como los procesos DataNode, NameNode y ResourceManager) se ejecutan en un solo servidor. (https://hadoop.apache.org/)

c. Totalmente distribuido.

   Real, configura Hadoop con parámetros óptimos con replicación de datos y posiblemente lograr alta disponibilidad, porque instalará Hadoop en un conjunto de servidores-nodos diferentes y no en un solo nodo. (https://hadoop.apache.org/)

2.1.13. Versiones Hadoop.

Hadoop 2, hay un único proceso global de ResourceManager que administra los recursos en todo el clúster y se ejecuta en un nodo maestro. Los nodos de trabajo tendrán todos NodeManagers similiar a TaskTracker. En Hadoop 1, JobTracker gestiona los trabajos, funciones de programación y administración de tareas. En Hadoop 2, las funciones de JobTracker se dividieron en dos: programación y administración de recursos, como referencia figura 11. (Alapati, 2017, pág. 22).

Tabla 2. Comparación de MapReduce 1 y MapReduce 2-YARN.

MapReduce 1	***MapReduce 2-YARN***
Jobtracker	ResourceManager, Application master
Tasktracker	NodeManager
Slot	Container

Nota: La diferencia radica el reemplazo de demonios JobTracker y Tasktracker. Por ResourceManager y NodeManager respectivamente. (White, 2013, pág. 101).

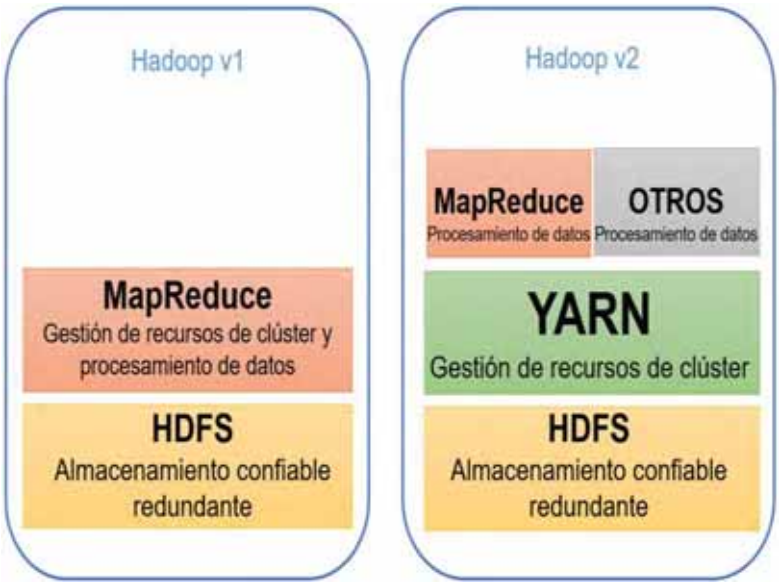


Figura 11. Versiones de Hadoop 1 y Hadoop 2 (White, 2013, pág. 101)

2.1.14. Modelo de Arquitectura Hadoop

Hadoop se basa en una arquitectura Master/Slave (Maestro/Esclavo) con tipos de nodos: nodo máster (maestro) y los nodos slave (esclavos). Un clúster Hadoop tiene un sólo nodo máster y varios nodos slave. (Mai, 2016, pág. 13).

2.1.15. Distribuciones Hadoop.

Empaquetado de tecnologías del ecosistema Hadoop con una capa de gestión añadida y soporte empresarial. Las principales son: Cloudera, Hortonworks., MapR. Microsoft azure, AWS y etc, referencia figura 12.

Figura 12. Distribuciones con licencia y sin licencia.

2.1.16. Rack

En Hadoop se denomina rack a la combinación de “nodos de datos”. Un rack puede tener máximo de 40 nodos máster, cada rack tiene un switch que le permite comunicarse con los distintos racks del ecosistema, sus nodos y procesos cliente, referencia figura 13. (Mai, 2016, pág. 14).

Figura 13. Rack de clúster Hadoop (Mai, 2016, pág. 15)

2.1.17. Contraste Sistema tradicional con MapReduce.

Comparación del Sistema Administración de Base de Datos Relacional con MapReduce, como referencia la tabla 3.

Tabla 3. DBMS comparado con MapReduce

Descripción	***RDBMS tradicional***	***MapReduce***
Tamaño de datos	GigaByte.	PetaByte.
Acceso.	Interactivo y por lotes.	Batch.
Actualizaciones.	Leer y escribir muchas veces.	Escribe una vez, lee muchas veces.
Transacciones.	A.C.I.D.	Ninguna.
Estructura.	Esquema de escritura	Escalabilidad horizontal.
Integración.	Alta.	Baja.
Escalabilidad	No lineal.	Lineal.

Nota: El contraste entre la RDBMS con MapReduce. (White, 2013, pág. 29)

2.1.18. HDFS-Sistema de Archivos Distribuido de Hadoop.

HDFS es que es un sistema de ficheros distribuido. Estos tipos de sistemas se usan para superar el límite que un disco duro individual. Cada máquina del clúster almacena un subconjunto de datos que componen el sistema de archivos completo con la idea de tener varias máquinas con distintos discos duros y así distribuir toda la información en ellos. Los metadatos se almacenan en un servidor centralizado actuando como un directorio de bloques y dotándolo de una visión global del estado del sistema de archivos. Otra diferencia con respecto al resto de sistemas de archivos es su tamaño de bloque. Es común que estos sistemas utilicen un tamaño de 4KB o 8KB para sus datos. Hadoop en cambio utiliza un tamaño de bloque significativamente mayor, 64MB por defecto, aunque los administradores de este tipo de clústers lo suelen elevar a 128MB o 256MB. El aumento del tamaño de los bloques provoca que los datos se escriban en trozos contiguos más grandes en el disco, que a su vez significa que se pueden escribir y leer en operaciones secuenciales más grandes. Esto minimiza la búsqueda de los bloques contiguos (es donde más tiempo se pierde), y mejora el rendimiento en operaciones de I/O. En lugar de recurrir a la protección de esos datos, HDFS replica cada bloque en varias máquinas del clúster (3 veces por defecto). Los cambios que se realizan en un bloque también se traspasan a sus réplicas, por lo que las aplicaciones pueden leer de cualquiera de los bloques disponibles. Tener múltiples réplicas significa tener más fallos, pero serán más fácilmente tolerados. HDFS rastrea activamente y gestiona el número de réplicas disponibles de un bloque, de forma que si el número de copias de uno de estos bloques está por debajo del factor de réplicas establecido, HDFS genera automáticamente una copia de las réplicas restantes. HDFS presenta el sistema

de ficheros como uno de alto-nivel con operaciones y conceptos familiares, como referencia figura 14. (Rodríguez, 2014, pág. 37).

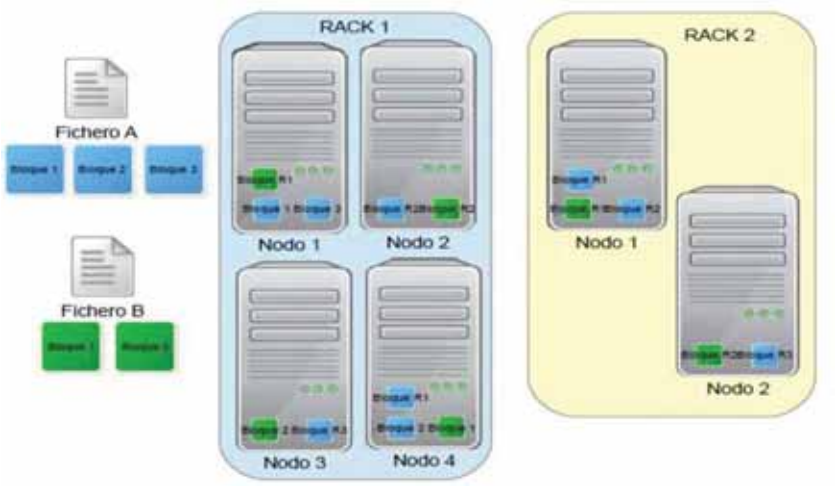


Figura 14. Ficheros a bloques y replicando en diferentes racks (Noelia, 2014)

2.1.19. Demonios de HDFS.

Hay tres tipos de demonios que conforman HDFS y cada uno tiene un papel distinto, se puede referenciar en la siguiente tabla 4.

Tabla 4. Demonios, números por clúster y funciones dentro del HDFS.

Demonio	***Número por clúster***	***Función***
NameNode	1	Almacena el metadato, los ficheros como un mapa de bloques y proporciona una imagen global del sistema de archivos.
Secondary NameNode	1	Actúa como una NameNode pasivo que almacena los checkpoints y log del NameNode activo.
DataNode.	1 o más.	Almacena los bloques de datos.

Nota: Son los demonios exclusivamente de HDFS.

Podemos visualizar la distribución HDFS los demonios, referencia figura 15.

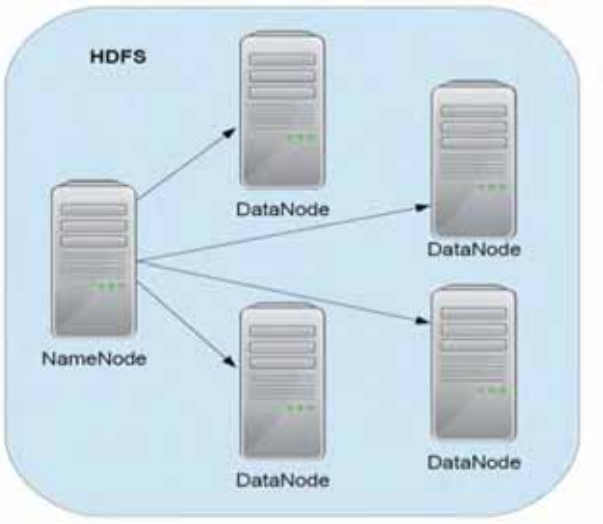


Figura 15. Arquitectura HDFS. (Noelia, 2014, pág. 45)

2.1.20. Sistemas distribuidos.

Serie de equipos independientes que actúan de forma transparente actuando como uno solo. Descentralizar el almacenamiento de información o servicios de procesamiento. Con mayor eficacia, velocidad, tolerancia a fallos y escalabilidad.

2.1.21. MapReduce.

Es procesar grandes cantidades (terabyte) de datos en paralelo y tolerante a fallos. El MapReduce y HDFS están normalmente en el mismo conjunto de nodos, lo que permite a Framework agendar tareas en los nodos que contienen datos. El nodo primario(master) agenda tareas, componentes de trabajo, trabajo de monitoreo y vuelve a ejecutar tareas falladas, y el nodo secundario (esclavo) ejecuta tareas según las indicaciones del nodo primario. (Noelia, 2014, pág. 40):

a. La fase Map.

   Es la primera parte de la secuencia de procesamiento de datos dentro de MapReduce. Las funciones map sirven de nodos que trabajan y son capaces de procesar varios fragmentos pequeños del conjunto de datos. Map es responsable de

dividir el conjunto de datos de entrada en trozos más pequeños, generando "clave/valor". (Noelia, 2014, pág. 40)

b. La fase Reduce.

Los pares "clave/valor" de las salidas de la función map, deben corresponder a la partición de reducción adecuada de manera que los resultados finales son agregados a datos correspondientes apropiadamente. Este proceso de mover las salidas de la función map hacia las funciones de reducción es conocida como "shuffling". Una vez que el proceso "shuffling" ha finalizado y los reductores (encargados de las funciones de reducción) ha copiado todas las salidas de la función de map, los reductores pueden entrar en lo que se conoce como un proceso de mezcla. Durante esta etapa de la fase de reducción, todas las salidas de la función map se pueden combinar juntas manteniendo su tipo de petición establecido durante la fase de map. La última tarea de reducción es consolidación de todos los resultados. Para cada clave dentro de la salida ya mezclada, y el resultado final se escriben en HDFS, referencia figura 16. (Noelia, 2014, pág. 40).

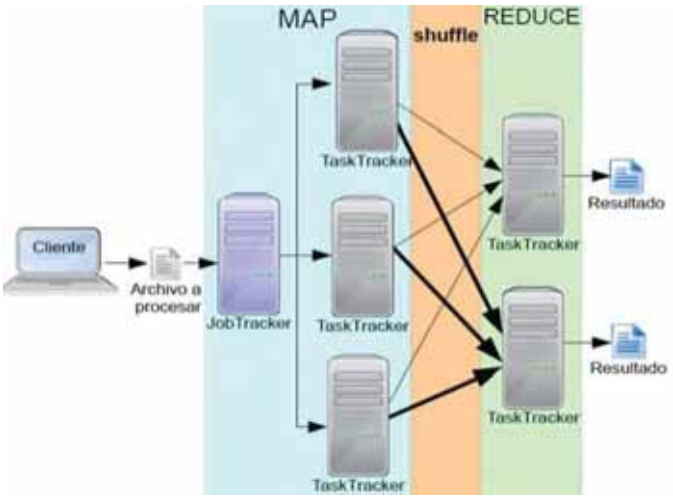


Figura 16. Arquitectura MapReduce. (Noelia, 2014, pág. 40)

2.1.22. YARN

Considerado sistema operativo del Ecosistema Hadoop, para así acoplarse a otros Frameworks. YARN se enfoca en gestionar recursos del clúster a la hora de realizar un proceso MapReduce. Hasta ahora el JobTracker realizaba dos funciones destacadas: la gestión de recursos, la planificación y monitorización de trabajos. YARN, reemplaza el demonio JobTracker por ResourceManger utiliza la abstracción para agrupar recursos de clúster como el disco duro, procesador, la memoria, la red, etc. (Echevarría, Estudio, Análisis y evaluación del Framework Hadoop, 2014, pág. 59)

a. ResourceManager.

   Servicio que se encarga de gestionar recursos del clúster. Tiene dos partes: (Echevarría, Estudio, análisis y evaluación del framework Hadoop, 2014, pág. 60)

   - Scheduler.

     Responsable de asignar los recursos a cada aplicación. Diseñado para tareas de planificación. (Echevarría, Estudio, Análisis y evaluación del Framework Hadoop, 2014, pág. 59)

   - ApplicationsManager.

     Responsable a peticiones de creación de trabajos y asigna ApplicationMaster a cada uno. (Echevarría, Estudio, Análisis y evaluación del Framework Hadoop, 2014, pág. 59)

b. ApplicationMaster.

   Servicio único para cada aplicación, su principal cometido es el de negociar con el ResourceManager la gestión de los recursos y la de trabajar con los NodeManagers para ejecutar y monitorizar los trabajos. (Echevarría, Estudio, Análisis y evaluación del Framework Hadoop, 2014, pág. 60)

c. NodeManager.

Este servicio o demonio se ejecuta en cada nodo (esclavo) y es responsable del contenedor e informa su estado a Scheduler. (Echevarría, Estudio, Análisis y evaluación del Framework Hadoop, 2014, pág. 60).

La figura 17, muestra el proceso YARN con la secuencia siguiente:

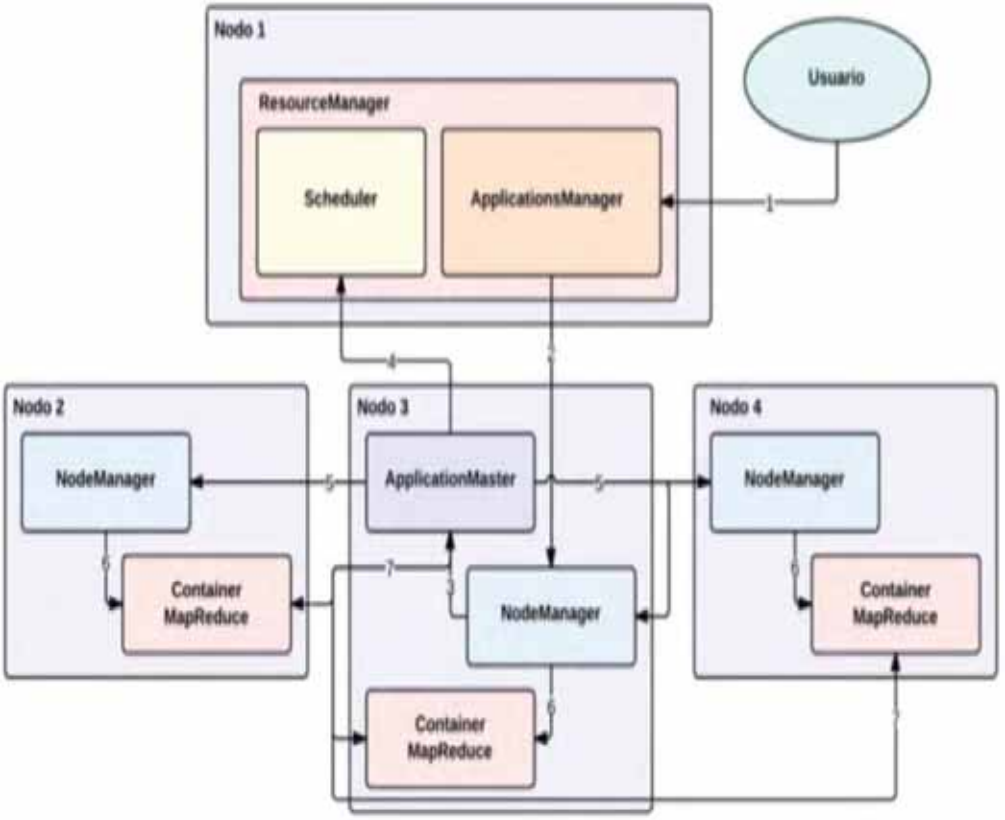


Figura 17. Funcionamiento de arquitectura YARN. (Echevarría, Estudio, Análisis y evaluación del Framework Hadoop, 2014, pág. 61)

2.2 Marco Conceptual (palabras clave)

2.2.1. Por lotes/batch.

Programado para su ejecución diferida en el tiempo. (https://www.capgemini.com/, pág. 19)

2.2.2. Casi Tiempo Real/Near real-time.

Existe una pequeña diferencia de tiempo entre el momento en que ocurre el evento y en el que se procesa (segundos, o pocos minutos). (https://www.capgemini.com/, pág. 19)

2.2.3. Tiempo Real/real-time.

La información se procesa y se consume en intervalos de tiempo muy muy pequeños (milisegundos, microsegundos). (https://www.capgemini.com/, pág. 19)

2.2.4. Escalabilidad vertical

Este tipo de escalamiento tiene aspectos negativos, porque este crecimiento está ligado a hardware, luego tarde o temprano tendrá un límite, llegará el momento de incrementar el mejor procesador, el mejor disco duro, la mejor memoria y no podamos crecer más o podríamos a lo mejor comprar el siguiente modelo de servidores que costará exorbitantemente. (https://www.oscarblancarteblog.com/)

2.2.5. Escalabilidad horizontal

Implica varios servidores-nodos trabajando como uno solo, se interconectan con la finalidad de repartirse el trabajo, cuando el performance del clúster se ve afectada se añaden nuevos nodos. (https://www.oscarblancarteblog.com/)

2.2.6. Tolerancia a fallos

Un fallo de hardware implica caída de su respectivo nodo, previo a ello hay réplicas de bloques en diferentes nodos y rack. (Echevarría, Estudio, análisis y evaluación del framework Hadoop, 2014, pág. 44)

2.2.7. Clúster.

Aplica a una serie de computadoras construidas mediante la utilización de hardware común y que se comportan como si fuese una única computadora. (https://es.wikipedia.org/wiki/Cluster), referencia figura 18.

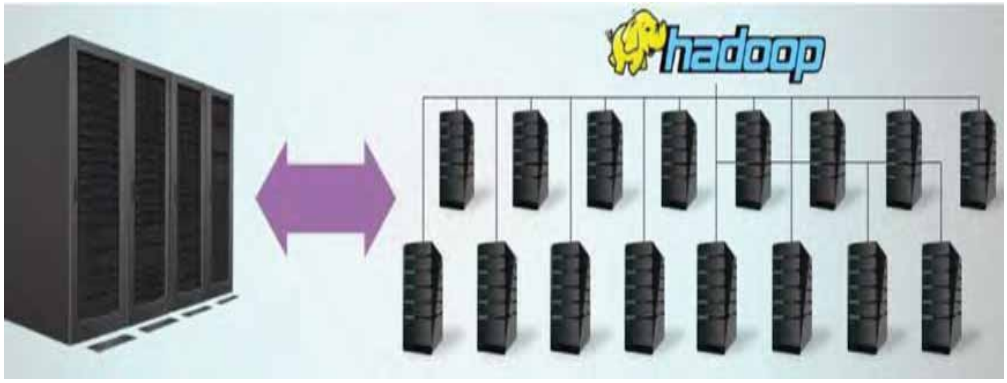


Figura 18. Clúster (https://www.capgemini.com/, pág. 6)

2.2.8. Replicación.

Un bloque replica en varios nodos en función del "factor de replicación" del clúster, que podría ser 1, 2 y 3

2.2.9. Curva de ajuste

Muy a menudo en la práctica se encuentra que existe una relación entre dos (o más) variables, y se desea expresar esta relación en forma matemática determinando una ecuación que conecte las variables.

Del diagrama de dispersión es posible frecuentemente visualizar una curva que se aproxime a los datos. Dicha curva se llama curva de aproximación. En la Fig. A, por ejemplo, se observa que los datos se aproximan bien por una recta y decimos que existe una relación lineal entre las variables. Sin embargo, en la Fig. B aunque existe una relación entre las variables. ésta no es una relación lineal y por esto la llamamos relación no lineal. En la Fig. C parece que no hay ninguna relación entre las variables. (Levin, 2004, pág. 600) referencia figura 19.

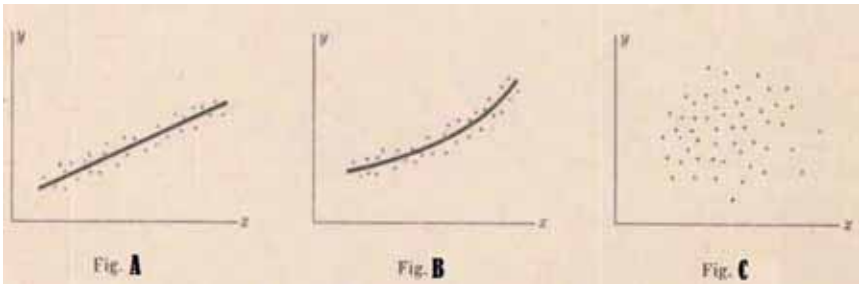


Figura 19. Figuras de curvas

2.2.10. Regresión lineal

Uno de los propósitos principales de la curva de ajuste es estimar una de las variables (la variable dependiente) de la otra (la variable independiente). El proceso de estimación se conoce como regresión. Si y se va a estimar a partir de x por medio de alguna ecuación la llamamos ecuación de regresión de y sobre x y a la curva correspondiente curva de regresión de y sobre x. (Levin, 2004, pág. 510)

2.3 Antecedentes empíricos de la investigación

2.3.1. Verma C, (2016), *"Representación de grandes datos para el análisis de grado a través de Hadoop Framework",* Universidad Amity, India.

Conclusión:

El sistema de estimación de leyes se basa en MapReduce arquitectura y framework basado en Hadoop. La propuesta la calificación de arquitectura del estudiante se puede utilizar para hacer predicación. Se puede utilizar para el análisis de varios atributos del marco de Hadoop sobre el entorno de nube. El papel ha deliberado claramente sobre la distribución de datos y las respectivas pares clave-valor en cada etapa de la arquitectura Hadoop.

Comentario:

Respalda a nuestro diseño de arquitectura con MapReduce y Hadoop que permite resolver problemas de grandes volúmenes de datos no estructurados en base a tupla (clave-valor). Refuerza la idea de contar con información que nos permitan estimar tiempos y rendimientos de clúster, así mismo aplicando funciones matemáticas que permitan estimar a priori.

Además en el 2011 Gartner definió “grandes datos como desafíos de crecimiento se centran en 3V el volumen, variedad y velocidad”, referencia figura 20. (Verma, Big Data representation for Grade Analysis Through, 2016).

` *Figura 20.. Mostrar las 3Vs en Big Data.*

Con Hadoop como marco de software para calcular gran cantidad de datos, se compone de cuatro módulos principales. Estos módulos son: Hadoop Common, Sistema de Archivos Distribuidos Hadoop (HDFS), Hadoop YARN, Hadoop MapReduce (Hadoop MapReduce divide el problema grande en sub problemas menores). referencia la figura 21. (Verma, Big Data representation for Grade Analysis Through, 2016)

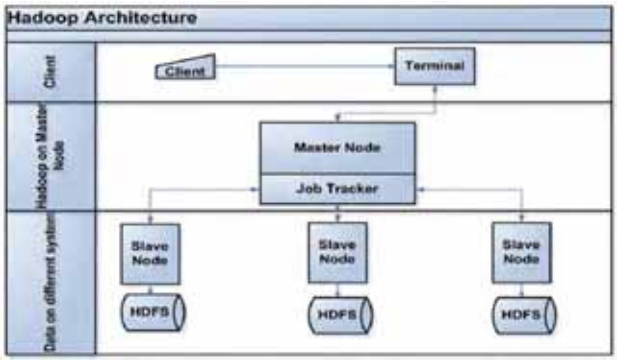


Figura 21. Hadoop y componentes (Verma, Big Data representation for Grade Analysis Through, 2016)

2.3.2. Kaur Sidhu, (2016), "*Procesamiento por lotes eficiente de tareas relacionadas de Big Data utilizando la técnica de MapReduce persistente"*, Universidad Punjabi, India.

Conclusión:

La técnica propuesta, es decir un script avanzado toma aproximadamente la mitad del tiempo que toma la simple técnica. De este modo, la mejora de la eficiencia en el procesamiento por lotes de tareas relacionadas deduciendo el tiempo de ejecución al cincuenta por ciento del tiempo que tomó el enfoque anterior.

Comentario:

Esta investigación da utilidad de mejorar la arquitectura en base a un script escrito en java, python y R. Que permite estimar tiempos y rendimientos de clúster homogénea y heterogénea, además se puede optimizar el MapReduce, al configurar los parámetros del framework Hadoop.

2.3.3. Rehan Ghazi, (2015),*"Hadoop, MapReduce y HDFS: Una perspectiva de desarrolladores",* Instituto de administración y tecnología Bhubaneswar, India.

Conclusión:

El paradigma de programación Hadoop MapReduce y HDFS se utilizan cada vez más para procesar conjuntos de datos grandes y no estructurados. Hadoop permite interactuar con el modelo de programación MapReduce mientras oculta

la complejidad de implementar, configurar y ejecutar los componentes de software en la nube pública o privada. Hadoop permite a los usuarios crear un clúster de servidores básicos. MapReduce se ha modelado como una capa de plataforma como servicio independiente adecuada para diferentes requisitos de los proveedores de la nube. También permite a los usuarios comprender el procesamiento y análisis de datos

Comentario:

Esta revista indica las utilidades de las 3 versiones de Hadoop, desde el framework Hadoop versión 1, érase la mejor alternativa de almacenamiento para datos no estructurados y ahora es optimizado con Hadoop versión 2 estable, con YARN es de uso actual y finalmente da inicio a Hadoop versión 3 versión alpha. Actualmente se aplica en cloud Dataproc de Google, AWS, etc.

Adicionando la arquitectura Hadoop de versión 1, como muestra la figura 22. (Ghazi, 2015).

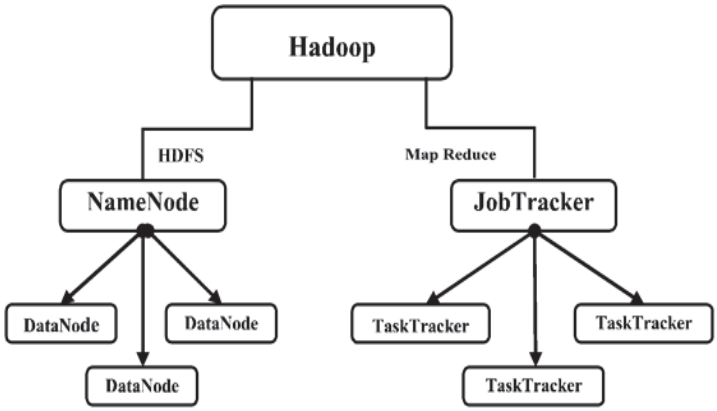


Figura 22. Arquitectura Hadoop (Ghazi, 2015).

HDFS, conlleva a un nodo llamado DataNode, divide los ficheros en bloques de tamaño (64, 128 MB) y el Secondary NameNode No es una copia de seguridad para

el NameNode: el trabajo NameNode secundario es leer periódicamente el sistema de archivos, referencia figura 23.

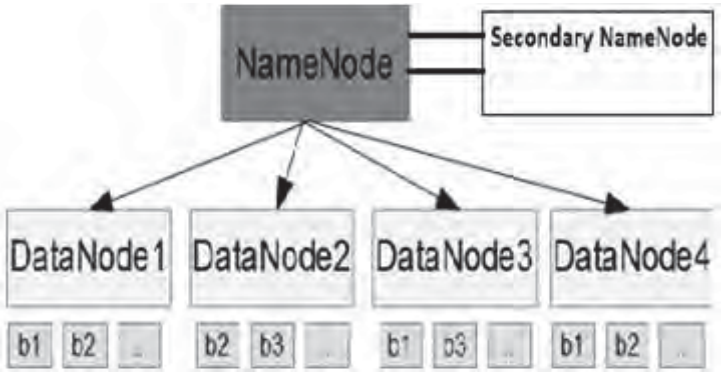


Figura 23. Arquitectura HDFS. (Ghazi, 2015)

MapReduce.- Es un modelo de programación que sirve para la manipular datos que se almacenan en un clúster de servidores conectados para procesamiento de paralelismo masivo, referencia figura 24.

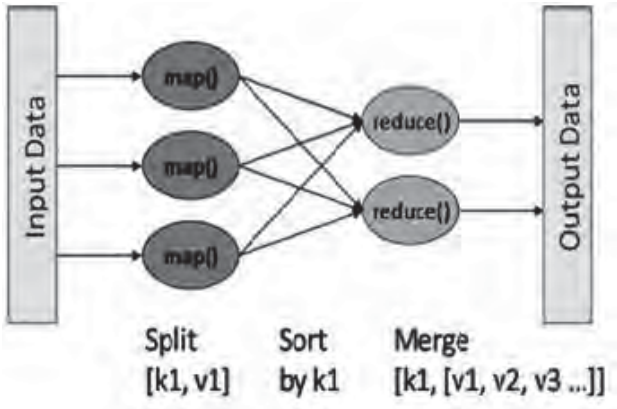


Figura 24. Fase de Map y Reduce (Ghazi, 2015)

2.3.4. Prabhu S, (2015), *"Mejora del rendimiento de Hadoop MapReduce Framework para analizar Big Data",* Instituto de Nitte, India.

Conclusión:

Consideramos los datos de Web Log para nuestro análisis y experimentó con los parámetros de configuración predeterminados de Hadoop que llamamos como nuestro sistema de línea de base. Hemos analizado del resultado, es decir, si

optimizamos el sistema Hadoop con parámetros de configuración, entonces podemos mejorar el rendimiento de sistema. Así que trabajamos en la optimización del parámetro basado en los recursos del sistema y la aplicación y discutimos por qué la configuración de Hadoop debe ser cambiado de su valor predeterminado a específico de la aplicación configuración. De los resultados de este experimento en general el rendimiento ha mejorado en un 32,97%. Por la corriente trabajo que consideramos pocos parámetros, el trabajo futuro debería centrarse en encontrar otros parámetros que reduzcan el rendimiento de Hadoop y también aumentar el tamaño del clúster con ambiente heterogéneo

Comentario:

Esta investigación da utilidad es ajustar más el parámetro de CPU y podamos estimar óptimamente los tiempos y rendimiento en el framework Hadoop, como muestra de trabajo un clúster de 3 nodos de hardware homogénea, cada nodo con características de procesador Intel Core i5 3470 CPU @ 3.20GHz * 4 de 3,8 GB de RAM, sistema operativo Ubuntu 14.04 con la versión estable Hadoop 1.2.1 utilizó oracle jdk 1.7, la configuración ssh y tamaño de bloque predeterminado 64 MB. Utilizando datos de log web de 2,1 GB, en consecuencia el tiempo total de la CPU es de 276 segundos en línea base y 231 segundos, dando una utilidad de propuesta a mejorar el rendimiento de la CPU en un 16,3%, referencia figura 25. (Prabhu, 2015).

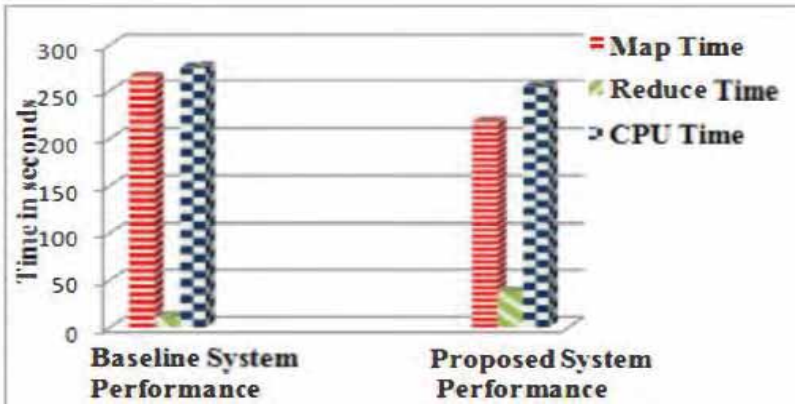


Figura 25. Tiempo Map, Reduce y CPU. (Prabhu, 2015)

2.3.5. Manikandan S, (2014),*"Análisis de datos grandes utilizando Apache Hadoop"*, Departamento de tecnología de información Tambaram, Chennai, India.

Conclusión:

La necesidad de procesar enormes cantidades de datos nunca ha sido mayor. No solo son escalas de terabytes y petabytes. Análisis de Big Data herramientas como MapReduce sobre Hadoop y HDFS, promete ayudar a las organizaciones a comprender mejor a sus clientes, lo que con suerte conducirá a mejores decisiones comerciales y ventajas competitivas. Para ingenieros de construcción herramientas y aplicaciones de procesamiento de información, grandes y conjuntos de datos heterogéneos que generan flujo de datos, conducen a algoritmos más efectivos.

Comentario:

Esta investigación respalda lo ineludible de almacenar datos de unidades de medida como: ZetaByte, YottaByte y BrontoByte en el framework de Hadoop. Para dicho almacenamiento, la mejor opción enmarcado como muestra la figura 26. (Manikandan, 2014)

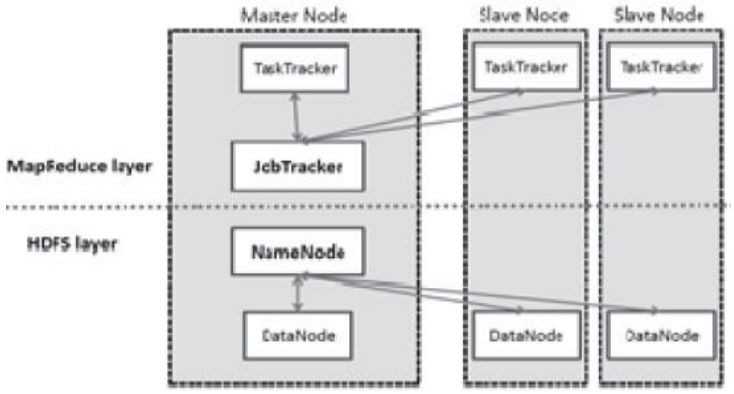


Figura 26. Arquitectura HDFS. (Manikandan, 2014)

Basándose en un procesamiento paralelo de los conjuntos de datos a gran escala con MapReduce enmarcado como muestra la figura 27. (Manikandan, 2014).

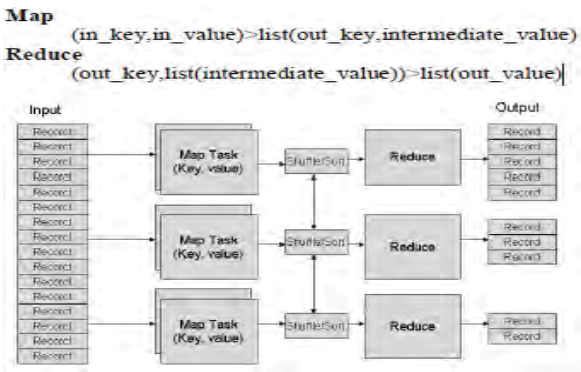


Figura 27. Arquitectura Map Reduce. (Manikandan, 2014)

2.3.6. Pal A, (2014),*"Enfoque experimental hacia grandes datos para analizar la utilización de memoria en un clúster Hadoop utilizando HDFS y MapReduce",* Instituto nacional de Bhopal, India.

Conclusión:

Este trabajo muestra el comportamiento del clúster Hadoop con número creciente de nodos. El parámetro para el comportamiento se analiza con los parámetros de la memoria. Este trabajo será útil para el desarrollo de un clúster homogéneo de Hadoop. La comunicación aumenta a medida que el tamaño del clúster aumenta. No recomendaremos el uso de la instalación de Wubi de Ubuntu

para desarrollar el clúster porque después de algún tiempo comenzará a dar problemas. Ubuntu puede ser muy eficiente para desarrollar el clúster hadoop. Si el tamaño de los datos aumenta y puede haber una posibilidad de disco, entonces el script de copia estándar debe usarse para aumentar el tamaño de los discos virtuales

Comentario:

Esta investigación es útil para el tratamiento distribuido de datos no estructurados, en clúster meramente de arquitectura homogénea basado en Hadoop. Configuradas con recurso de utilización en la tabla 5 y figura 28. (Pal, 2014)

Tabla 5. Test del experimento Hadoop.

System	***Dell***
RAM.	4 GB.
Disk.	30 GB.
Procesador.	Intel(R)Xeon(R) CPU W3565@3.20. GHz.
CPU.	64 bits
Operating System.	Ubuntu 12.04
Hadoop.	Hadoop-1.2-1-bin.tar.
Java.	Java OpenJDK 6
IP	Class B address.

Nota: El Test de prueba. (Pal, 2014)

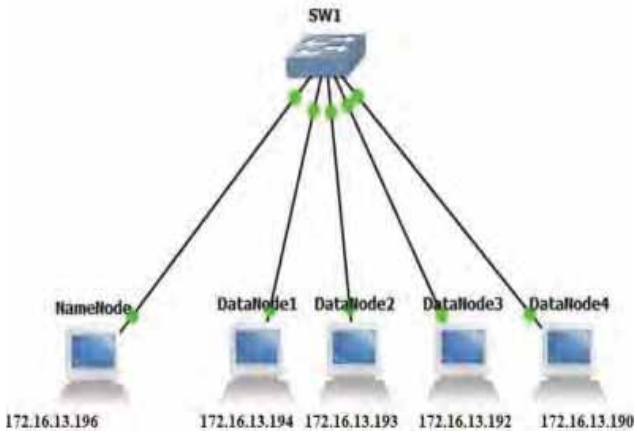


Figura 28. Distribución del clúster homogéneo. (Pal, 2014)

2.3.7. Mamta Padole, (2018), **"***Equilibrio de carga a través de la política de reordenación de bloques para el clúster heterogéneo de Hadoop",* Universidad de Baroda, India.

Conclusión:

Proponemos una política de colocación de bloques personalizada que aprovecha la capacidad de procesamiento de la CPU, durante el bloque colocación. Este enfoque será útil en MapReduce para minimizar el entrenudo y la transferencia entre bastidores. Tenemos demostró que podemos colocar bloques de datos de un archivo específico solo a nodos específicos. Este enfoque no afectará al equilibrio de carga general del clúster, ya que el resto de los archivos no se verán afectados. Los resultados experimentales demuestran que este esquema puede ser aplicado tanto a heterogéneos como a homogéneos grupos.

Comentario:

Esta investigación respalda a diseñar una arquitectura de especificaciones heterogéneas, para poder estimar la correlación las variables de tiempos de rendimientos basados en la cantidad de nodos, que se incrementan escalarmente en el clúster.

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Tipo y Diseño de investigación.

La investigación a realizar es cuantitativa, con alcance correlacional, que asocian conceptos, permiten predicción y cuantifican relació entre conceptos o variables. (Hernández-Sampieri, 2014).

3.2 Unidades de análisis.

Las unidades de análisis incorporado en el presente proyecto serán los datos no estructurados, representados en libros en unidades de medida de GigaByte.

3.3 Población.

La población que se utilizará en el presente trabajo son los datos no estructurados de internet, del repositorio https://www.gutenberg.org/. Se basa en datos no estructurados y, se optó por unos libros digitales, cuyo peso original es de 6.4 GB, adicionalmente se amplió a 12.8 GB.

3.4 Tamaño de muestra.

La parte experimental se efectuará en un clúster de computadoras, por tanto, se puede trabajar con la totalidad de los datos de la población. En consecuencia, no se delimitará ni definirá un tamaño de muestra.

3.5 Técnicas de recolección de datos e información.

Para recolectar los tamaños de muestras, necesitamos herramientas informáticas como internet, PC, disco externo y, proceder a descargar la información https://www.gutenberg.org/, al clúster y apilar en dataset para la test.

3.6 Análisis e interpretación de la información.

Para el análisis y la interpretación de información del presente trabajo será en base a los resultados que se obtengan de los tiempos de ejecución realizado en clúster de Hadoop, el cual deberá cumplir con el objetivo de distribuir los datos no estructurados, que cumpla dicha tarea de forma satisfactoria, es decir con una fiabilidad de aciertos en los test.

CAPÍTULO IV

DISEÑO DE ARQUITECTURAS.

4.1.Infraestructura homogénea de Hadoop.

La infografía muestra 10 nodos, con características homogéneas, el primer nodo es maestro y los 9 nodos son esclavos, como referencia figura 29.

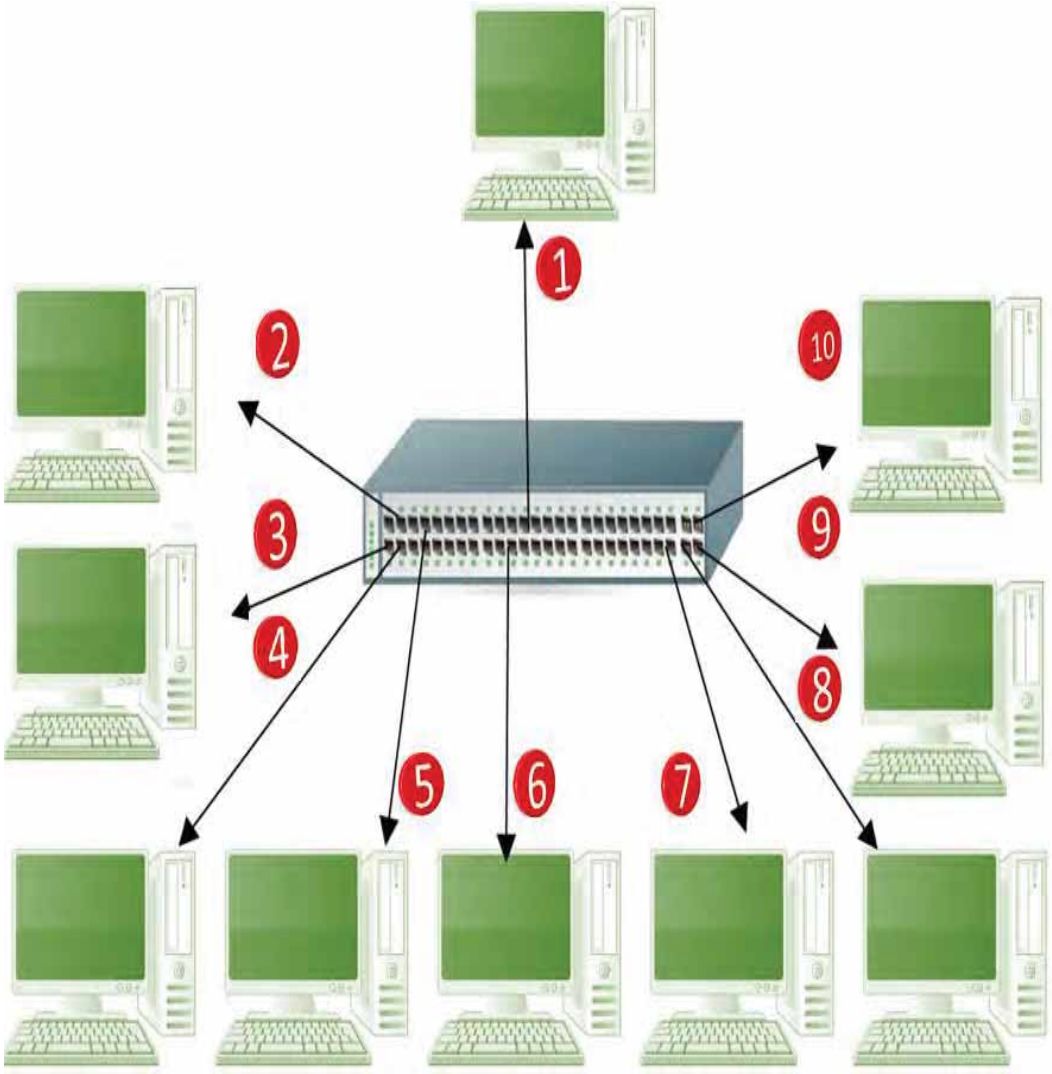


Figura 29. Clúster homogéneo (fuente propia).

Las características se detallan de los 10 nodos homogéneos, como referencia la tabla 6.

Tabla 6. RDBMS comparado con MapReduce

Descripción	*Características*
Sistema	Lenovo.
RAM.	16 GB.
Disco HDD.	Intel(R)Xeon(R)CPU E3-1270 v6@3.8Ghz 3.79GHz.
CPU.	64 bits.
Sistema Operativo.	Centos 7.
Hadoop.	Versión 2.9.2
Java	Version JDK 1.8.
Dirección IP	Clase C.

Nota: Característica de cada nodo de clúster homogéneo.

4.2.Instalación en un nodo

Base de la infraestructura para el diseño del clúster, se requiere descargar Hadoop 2.9.2.tar.gz de http://hadoop.apache.org/, jdk-8u241-linux-x64.rpm y, como referencia figura 30.

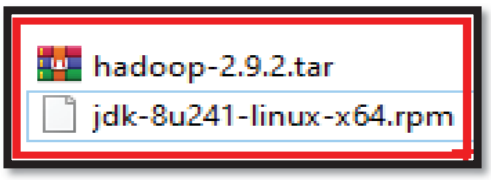


Figura 30. Ubicación de Hadoop, para descomprimir.

Instalar Hadoop en la unidad /opt –porque proporciona una ubicación donde instalar aplicaciones opcionales (de terceros). Creamos un directorio, pero con el root (súper usuario), ingresamos con su – root y tenemos el símbolo #, como referencia figura 31.

```
[root@localhost opt]# mkdir hadoop
[root@localhost opt]# chown hadoop hadoop
[root@localhost opt]#
```

Figura 31. Crear carpeta y cambiar el propietario.

Ahora extraemos con el comando tar xvf /home/hadoop/Descargas/Hadoop-2.9.2.tar.gz. Enseguida movemos Hadoop-2.9.2 a opt/Hadoop con el comando mv. Instalar JDK, la necesidad de instalar JDK para ejecutar javac (compilador de java), como referencia figura 32.

```
[root@localhost ~]# cd /home/hadoop/Descargas
[root@localhost Descargas]# ls -l
total 375076
-rw-r--r--. 1 hadoop hadoop 214092195 may  8 05:48 hadoop-2.7.3.tar.gz
-rw-r--r--. 1 hadoop hadoop 169983496 nov  4 01:54 jdk-8u131-linux-x64.rpm
[root@localhost Descargas]# rpm -ivh jdk-8u131-linux-x64.rpm
```

Figura 32. Ejecutar JDK para CentOS 7.

Contenido de Hadoop. El directorio /bin, contiene los binarios, ejecutables de Hadoop el directorio /etc, contiene ficheros de configuración de Hadoop, el directorio /sbin, contiene script de arranque, detener HDFS, YARN de Hadoop, el directorio /share, contiene algunos ejemplos y todo el core de Hadoop como primero el paquete se extrae, segundo el contenido de Hadoop, referencia figura 33.

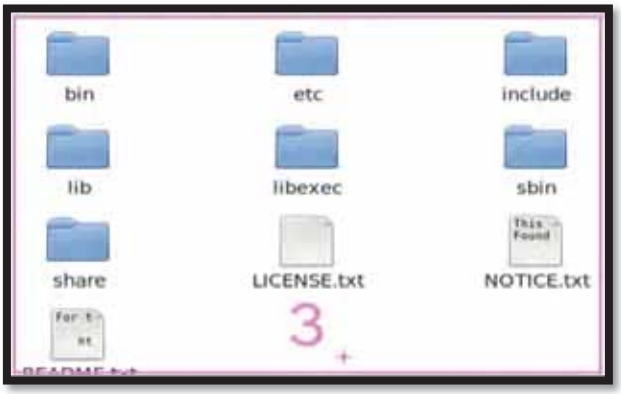


Figura 33. Contenido de Hadoop

Configuración JAVA. Nuestro primer mensaje de configurar la variable de entorno es error, como referencia figura 34.

Figura 34. Mensaje de error JAVA.

Para corregir el problema, creamos un fichero denominado **entorno.sh** con gedit, como referencia figura 35.

Figura 35. Corregir la configuración del path de JAVA.

Configurar SSH.La denominación es un Secure SHell, es la forma de que podamos ejecutarnos de manera segura en una administración remota de un clúster, para ello con el comando ssh-keygen generando dos llaves pública y privada como referencia figura 36.

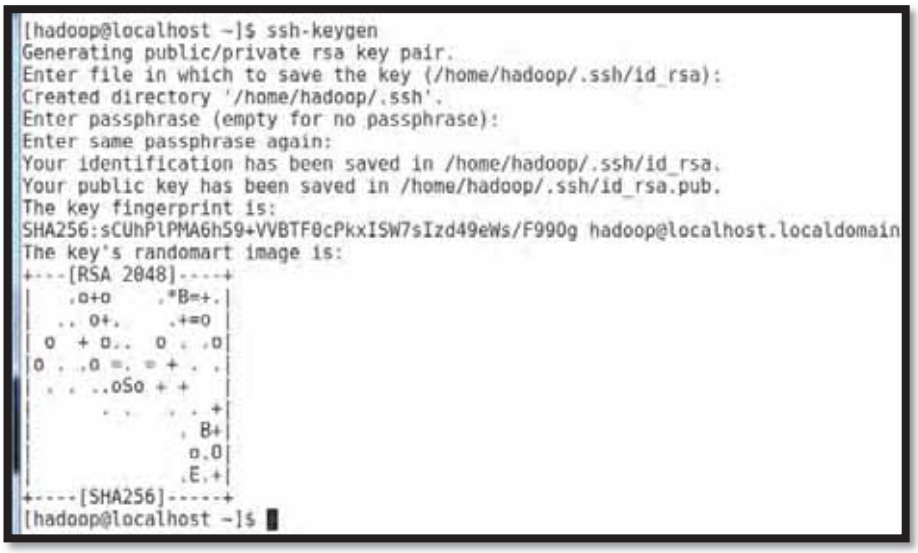


Figura 36. Ssh-keygen, nos permite generar clave pública y clave privada de cada nodo.

Generando el código de encriptación de la llave publica, como referencia la figura 37.

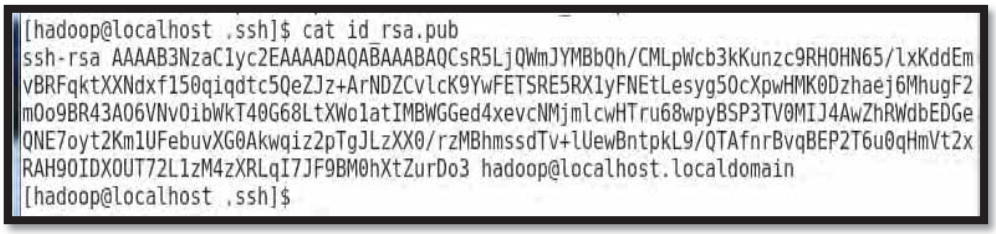


Figura 37.Clave pública.

4.3.HDFS en un nodo.

4.3.1. Fichero Core-site.xml

Configuración del clúster en el directorio **/opt/hadoop/etc/hadoop,** editando el fichero **core-site.xml**, como referencia figura 38.

Figura 38. Configurar fichero core-site.xml.

4.3.2. Fichero HDFS-site.xml

Configuración del clúster en el directorio **/opt/hadoop/etc/hadoop,** editando el fichero **hdfs-site.xml**, como referencia figura 39.

Figura 39. Configurar fichero hdfs-site.xml.

4.3.3. Crear directorio para maestro y esclavos.

Crear namenode y datanode del clúster, como referencia figura 40.

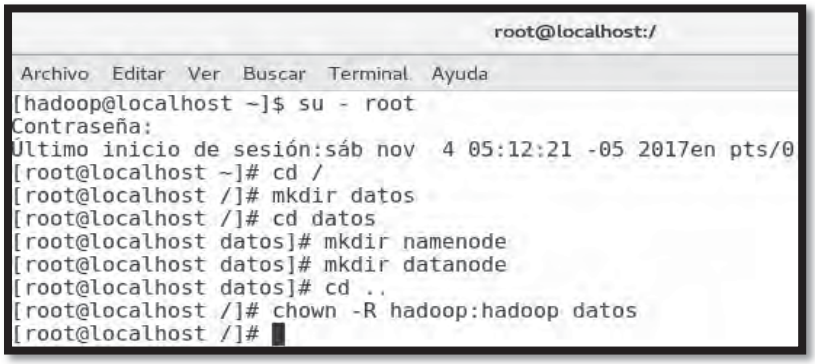


Figura 40. Comando mkdir como super usuario

4.3.4. Formatear Hadoop.

Pero antes de arrancar el clúster, ubicar el sistema de fichero Hadoop, con el comando **hadoop namenode -format** del nodo local, como referencia figura 41.

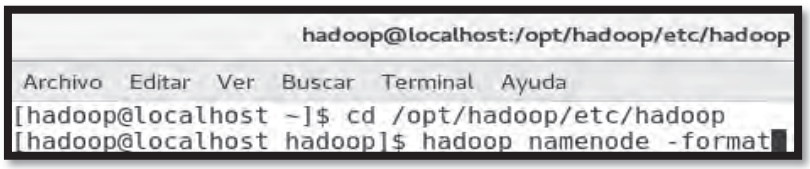


Figura 41. Formateamos al Hadoop.

4.3.5. Arrancar HDFS

En el sistema de directorio **/sbin** se encuentra el script de arranque de HDFS, entonces ahora ejecutamos el comando **start-dfs.sh** de HDFS, incluido el namenode y datanode y secondarynamenode, como referencia la figura 42.

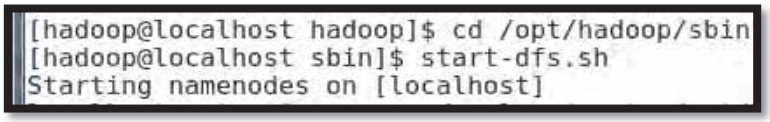


Figura 42. Iniciamos el HDFS

.

Confirmar los procesos java que tenemos ejecutando, dentro del jdk se encuentra con el comando **jps,** referencia de figura 43.

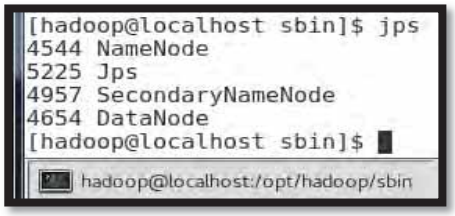


Figura 43. Confirmar la ejecución con JPS.

4.3.6. Mostrar Web de administración de HDFS.

Accedemos al modo web Administrativo de Hadoop por el puerto 50070, en el navegador de su preferencia, como referencia la figura 44.

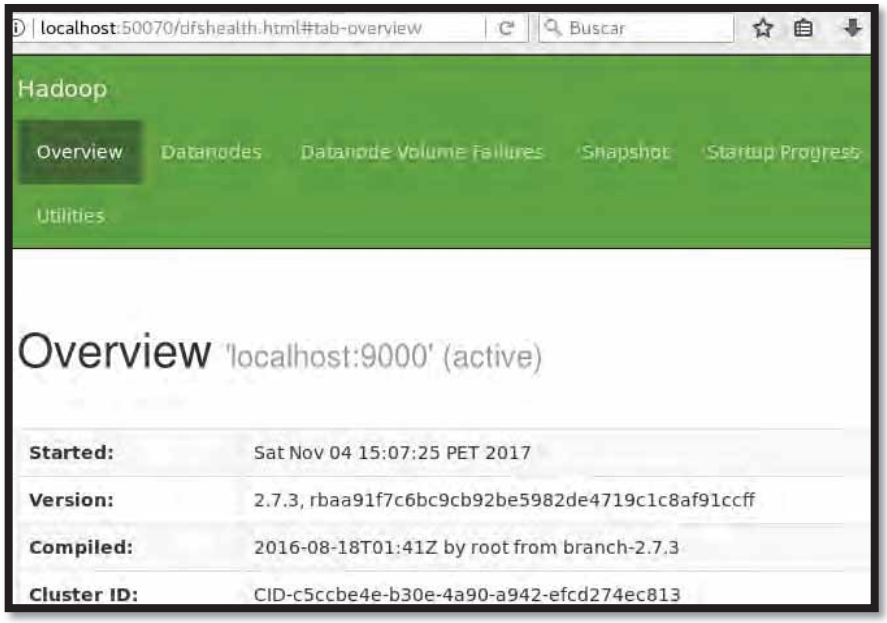


Figura 44. Confirmar la ejecución con JPS.

4.3.7. Comandos relevantes HDFS

Para visualizar cuantas réplicas de bloques hay en nuestro almacenamiento de HDFS del clúster, con el comando **hdfs dfsadmin–report**, como referencia la figura 45.

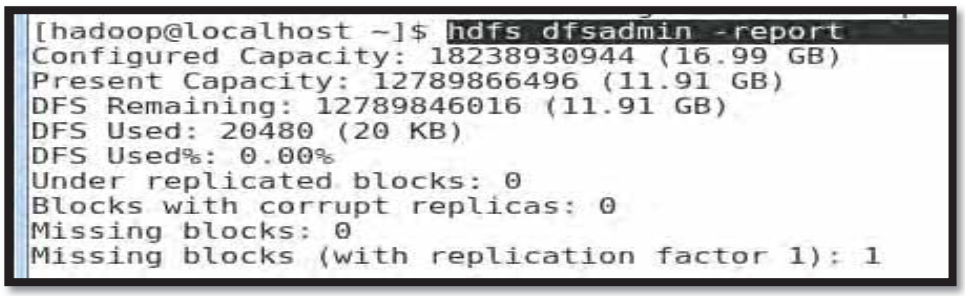


Figura 45. Comando para reportar las réplicas.

Para visualizar en compendio estado del clúster, si hay algún problema con las réplicas, lo pone solo lectura con el comando **hdfs fsck /** como referencia la figura 46.

Figura 46. Visualización de problemas de réplicas.

4.4.MapReduce & YARN en un nodo.

El modelo MapReduce se basa en 6 fases estas son: primero la entrada, cada nodo cargaría los bloques con los datos que tuviesen en su sistema de ficheros HDFS localmente. Segundo el split, obtiene una unidad de trabajo, par clave-valor, que comprende una sola tarea Map. Tercero Map, procesa los pares y produce un conjunto

de pares intermedio. Cuarto shuffle y sort, estos resultados intermedios (pares) son agrupados y ordenados por clave. Quinto reduce, esta obtención ordenada de pares en clave es aún procesados por otra serie de tareas denominado REDUCE para producir el resultado, como referencia la figura 47.

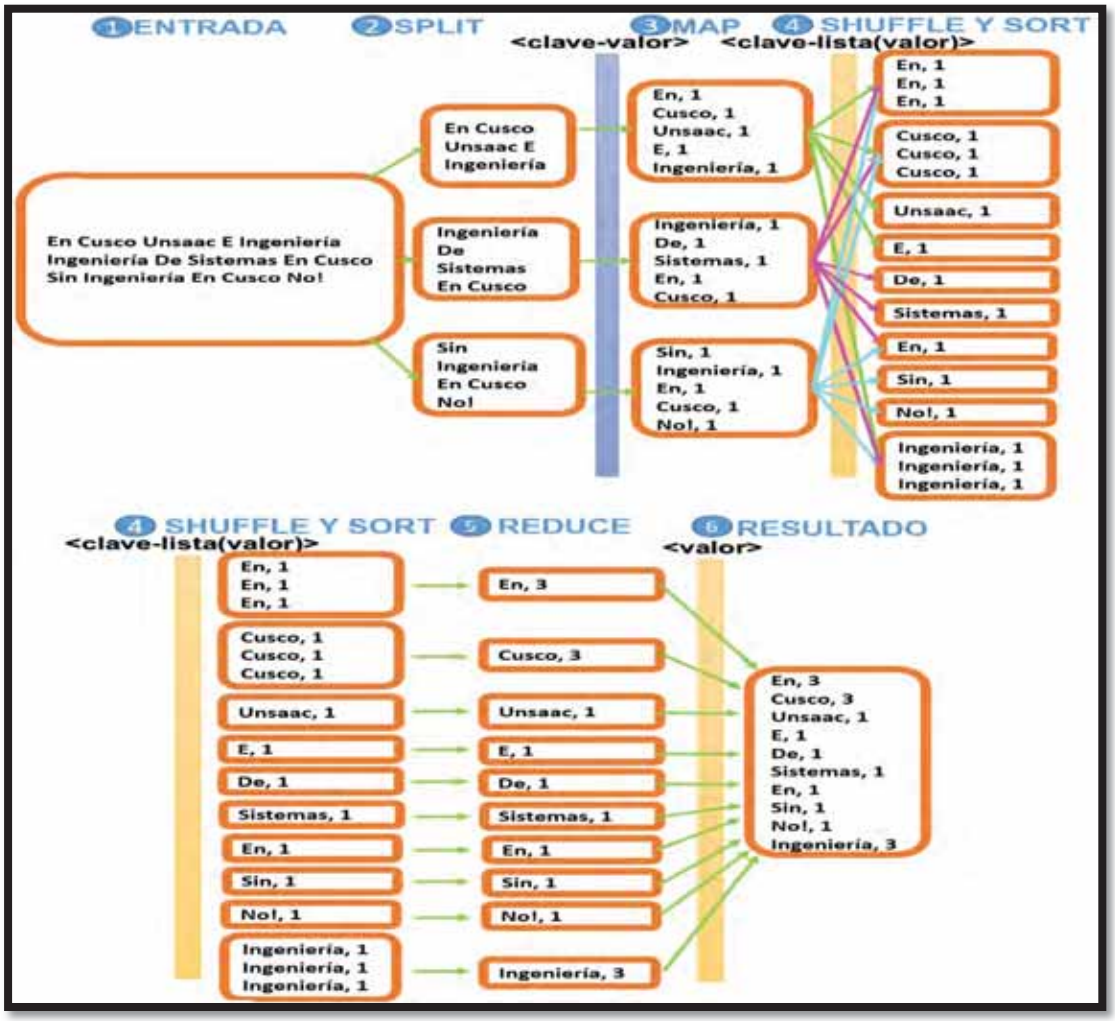


Figura 47. Ejecución lógica del procesamiento Hadoop versión 2.

Configurar YARN. ahora necesitamos el archivo **mapred-site.xml.template.** como plantilla para configurar, como referencia la figura 48.

```
<configuration>
                <property>
                        <name>mapreduce.framework.name</name>
                        <value>yarn</value>
                </property>
</configuration>
```

Figura 48. Configurar MapReduce a YARN.

El siguiente fichero a editar es con el contenido siguiente de 3 propiedades como son:

a. El gestor del manager y el nombre del host manager
b. La aplicación manager, de servicios
c. Y que clase manager, como referencia la figura 49.

```
<configuration>

<!-- Site specific YARN configuration properties -->
                <property>
                        <name>yarn.resourcemanager.hostname</name>
                        <value>nodo1</value>
                </property>
                <property>
                        <name>yarn.nodemanager.aux-services</name>
                        <value>mapreduce_shuffle</value>
                </property>
                <property>
                        <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
                        <value>org.apache.hadoop.mapreduce.ShuffleHandler</value>
                </property>
</configuration>
```

Figura 49. Configurar YARN.

Arrancar el YARN. Primero arrancamos con el HDFS, con el comando start-dfs.sh, los demonios a encontrar son, como referencia la figura 50:

```
3457 Jps
3079 DataNode
3271 SecondaryNameNode
2958 NameNode
```

Figura 50. Los demonios de HDFS.

Segundo con el comando **start-yarn.sh** iniciamos los proceso java de YARN, Levantado todos los procesos java o demonios a encontrar en un solo nodo maestro con el comando jps, como referencia figura 51.

```
[hadoop@localhost ~]$ jps
3616 ResourceManager
3079 DataNode
3271 SecondaryNameNode
3932 Jps
2958 NameNode
3743 NodeManager
[hadoop@localhost ~]$
```

Figura 51. Los demonios del clúster en un solo nodo.

Inicio de sesión del nodo configurado, como referencia la figura 52.

Figura 52. Sesión de inicio.

Proceder a realizar en todos los 10 nodos homogéneas, el mismo proceso.

4.5.Montar 10 nodos homogéneos.

Ingresar como super usuario para realizar cambios de configuración a ***nodo2***, como referencia la figura 53.

```
[hadoop@localhost ~]$ su - root
Contraseña:
Ultimo inicio de sesión:sáb nov  4 05:24:26 -05 2017en pts/0
[root@localhost ~]# hostname nodo2
[root@localhost ~]# uname -a
Linux nodo2 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27
 64 GNU/Linux
```

*Figura 53. Cambiamos el nombre a **nodo2***

Continuamos con la configuración de la red, ingresando a la carpeta /etc y sysconfig y finalmente editamos la red, se recalca como super usuario, como referencia la figura 54.

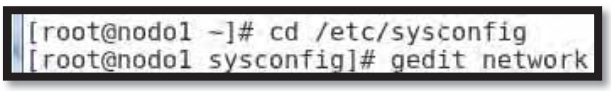


Figura 54. Editar en el directorio etc/sysconfig

El archivo de network, modificamos el nombre del host por ***nodo2***, como referencia la figura 55.

Figura 55. Cambiar de nombre al nodo.

Los nodos tienen que interactuar, entonces configuramos en cada nodo las **ip** verificando con el comando **ifconfig**, con el siguiente comando **gedit** hosts, editamos lo siguiente, recalcamos en todos los nodos, como referencia figura 56.

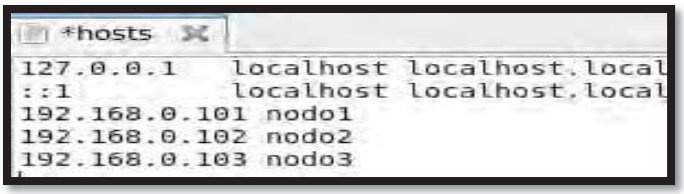


Figura 56. IP de los nodos.

4.6.Configurar SSH entre los 10 nodos

Realizamos la configuración de la red pública segura con .ssh. Ubicamos el fichero .ssh, volver a generar la autorización entre los 10 nodos con el comando **ssh-keygen** y presionamos 3 veces enter sin ingresar clave, como referencia la figura 57.

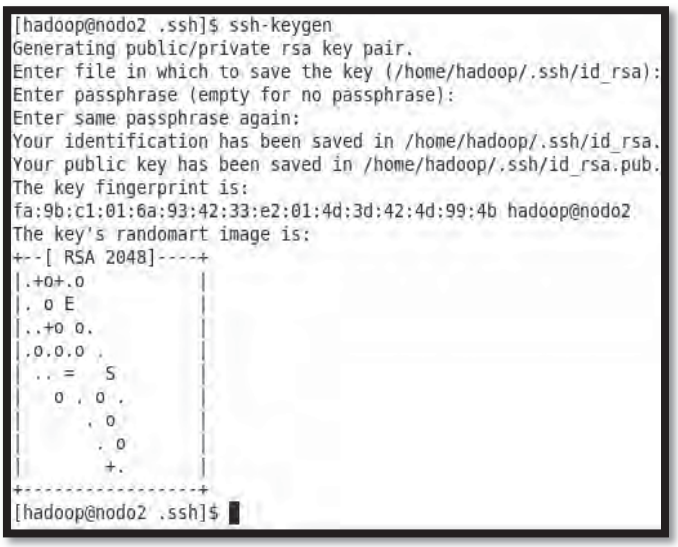


Figura 57. Generar un Nuevo SSH para el clúster.

4.7.Comunicación entre 10 nodos, modo ssh.

Configuramos y comprobamos la comunicación de los nodos del clúster con copiar el fichero generado con comando **cp id_rsa.pub authorized_keys**, enseguida pasamos el fichero de autorización del nodo 2 al nodo1 y nodo3 con el comando **scp authorized_keys** y finalmente adicionar la nueva autorización, con el comando **cat id_rsa.pub >> authorized_keys.** Como referencia de logueo de modo seguro en la figura 58.

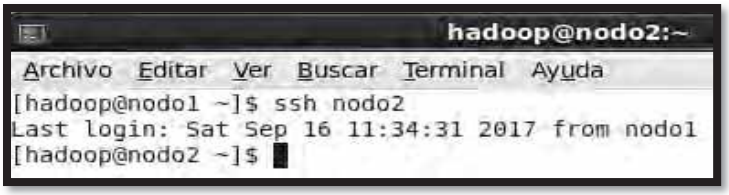


Figura 58. Ahora desde el nodo1, iniciar sesión con ssh al nodo2

4.8.Configurar un nodo maestro y 9 nodos esclavos

En sus respectivos ficheros **core-site.xml** editamos puertos y nombre del YARN que va corresponder **nodomaestro** en reemplazo a **localhost**, ubicándonos en el directorio de configuración etc. ahora configuramos lo ficheros de maestro y esclavos, como

referencia la figura 84 y configurar los datos en el fichero **hdfs.site.xml** y su NameNode y cantidad de réplicas del clúster a 2.

Indicar en el fichero **slaves,** los 9 nodos (nodo1, nodo2, nodo3, nodo4, nodo5, nodo6, nodo7, nodo8 y nodo9) esclavos, en nodo1 (maestro).

4.9.Arrancar 10 nodos homogéneos

Inicialmente iniciamos Hadoop en un solo nodo, ahora procedemos a iniciar el clúster distribuido, formateando con el comando **hdfs namenode–format,** como referencia la figura 59.

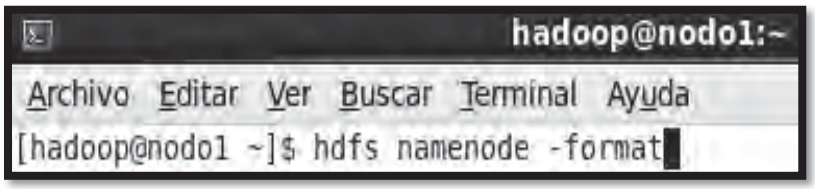


Figura 59. Iniciar el clúster primero formatear en namenode.

Continuando con el inicio del clúster de 10 nodos homogéneos, debemos levantar primeramente HDFS con el comando **start-dfs.sh, c**omprobamos los demonios en el nodomaestro con JPS, como referencia la figura 60.

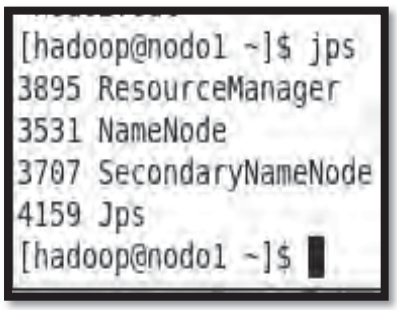


Figura 60. Demonios en el nodo Maestro.

En los nodos esclavos nodo1 y nodo2, comprobamos los demonios, como referencia figura 61.

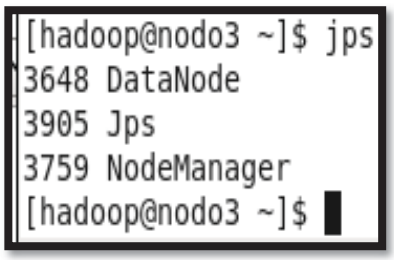


Figura 61. Demonios en todos los nodos esclavos.

Corroboramos el nodomaestro está activo, para administrar las aplicaciones MapReduce, así como los directorios de HDFS, como referencia figura 62.

Figura 62. Corroboramos el nodomaestro como activo.

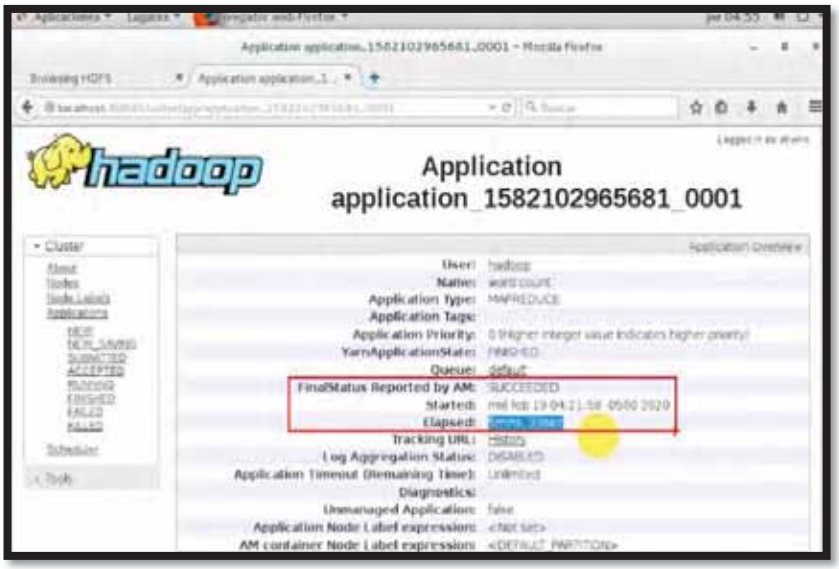


Figura 63. Clúster homogéneo - tiempo de respuesta con nodomaestro

Adicionar 2 nodos esclavos, los nodo1 y nodo2 son los nodos del diseño propuesto y desarrollado, como referencia figura 63.

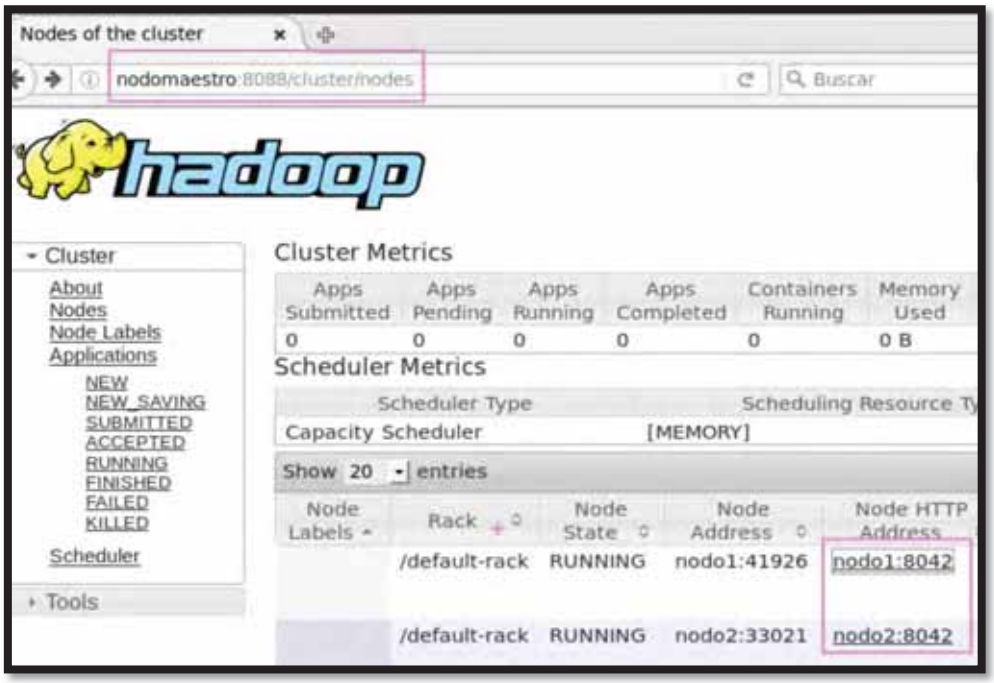


Figura 64. Clúster homogéneo - nodo1 y nodo2 son los esclavos del clúster.

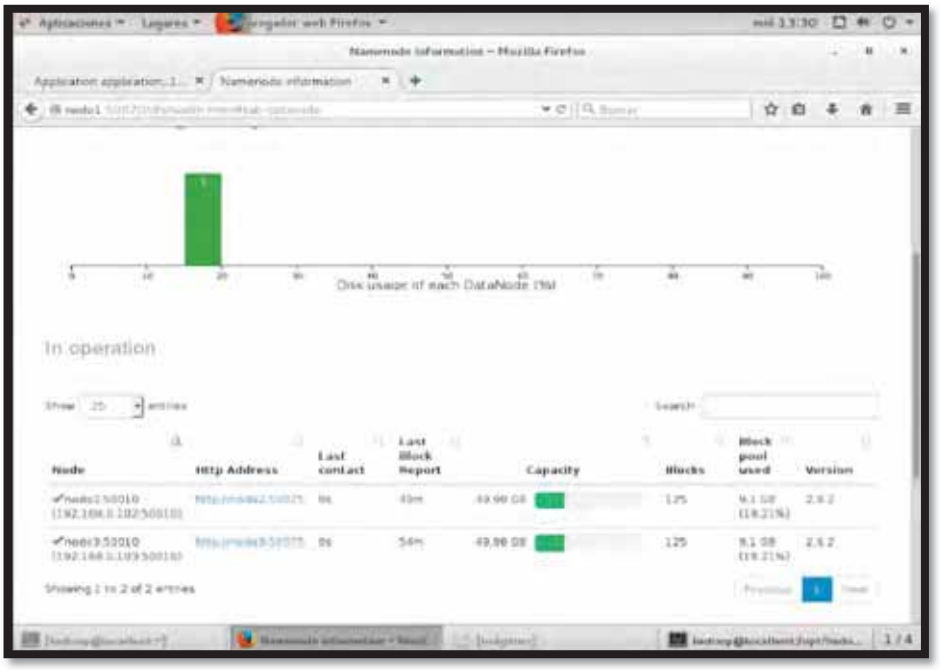


Figura 65. Clúster homogéneo - 2 nodos mostrados en NameNode

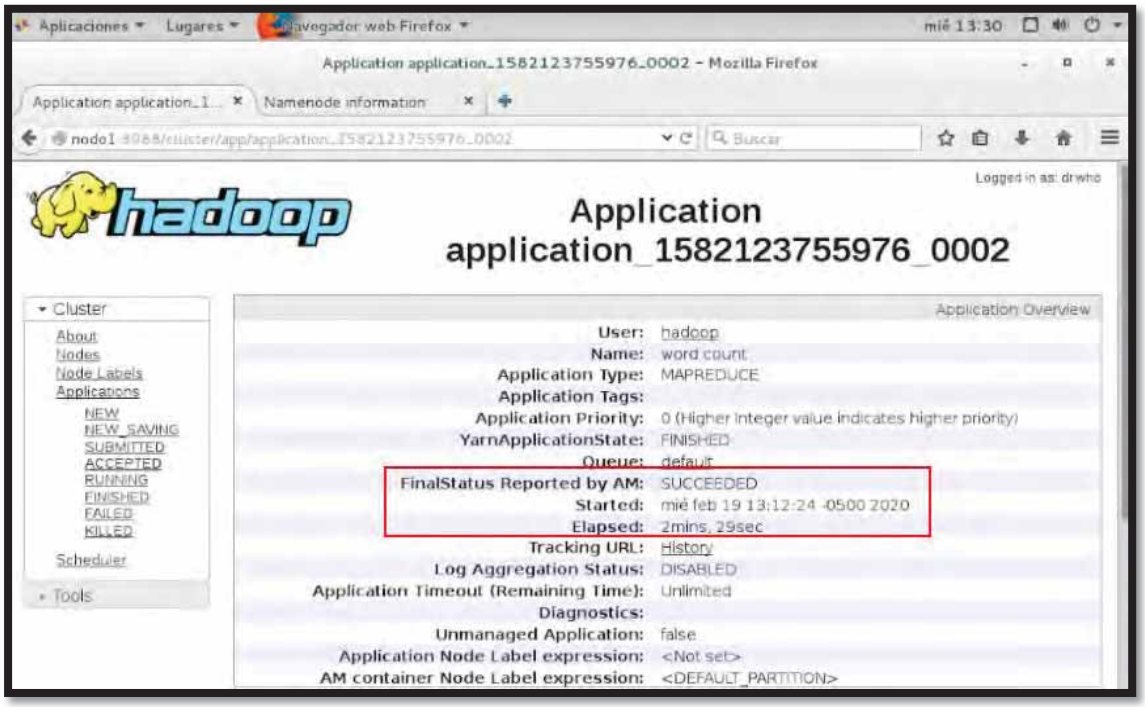


Figura 66. Clúster homogéneo - tiempo de respuesta con 3 nodos: 1 NameNode y 2 DataNodes

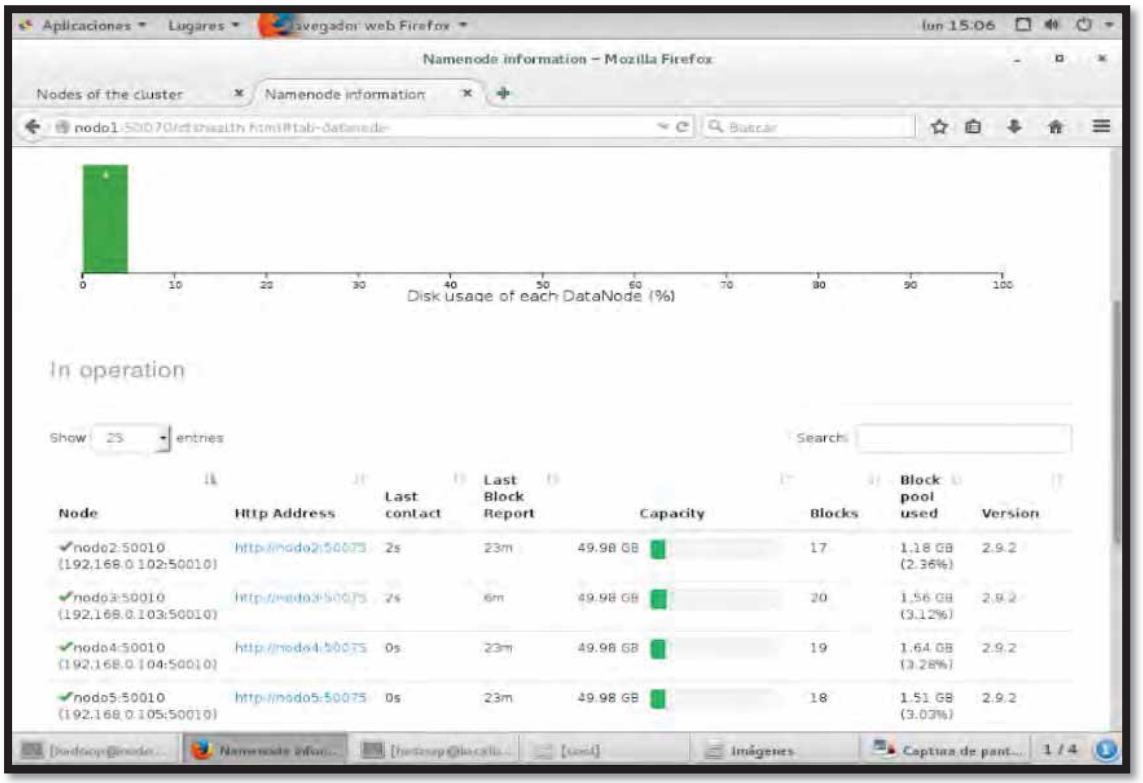


Figura 67. Clúster homogéneo - 4 nodos mostrados en NameNode

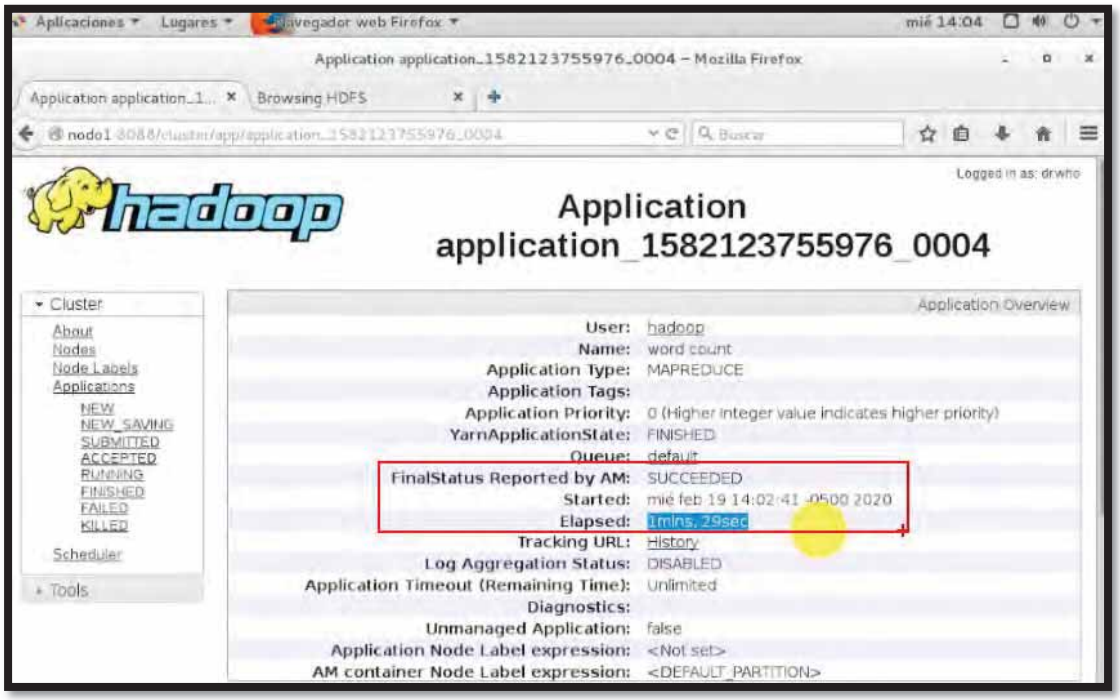


Figura 68. Clúster homogéneo - tiempo de respuesta con 5 nodos: 1 NameNode y 4 DataNodes

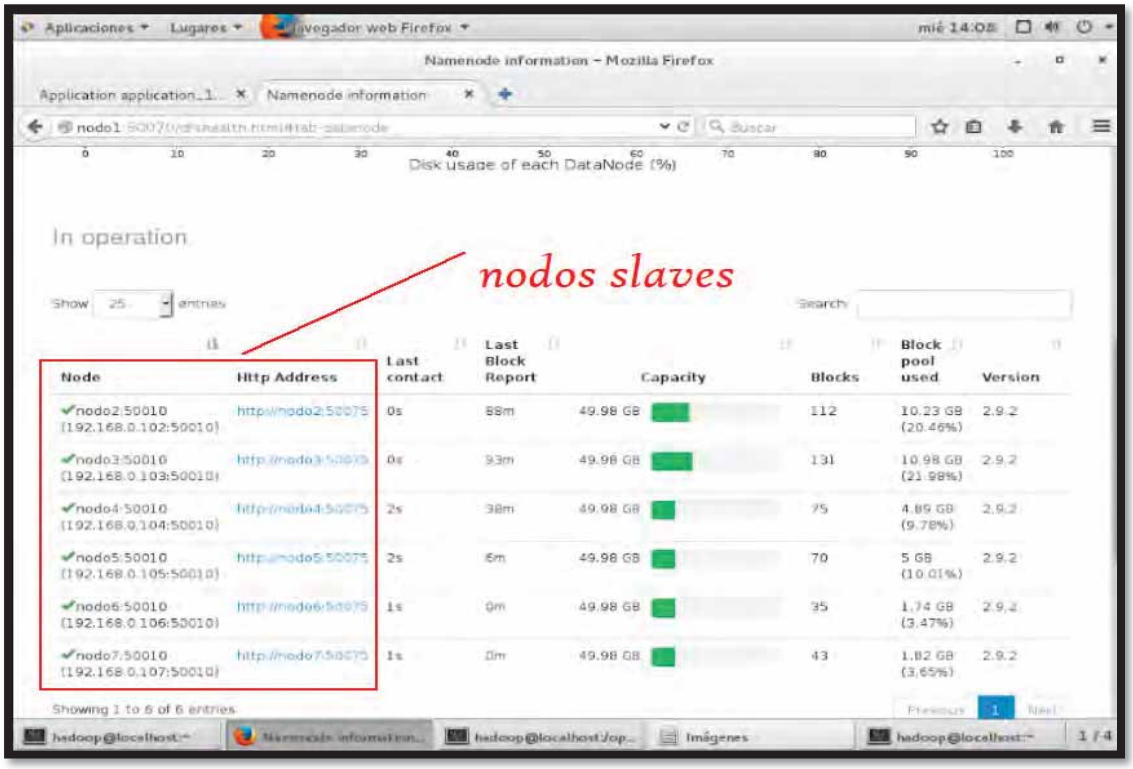


Figura 69. Clúster homogéneo - 6 nodos mostrados en NameNode

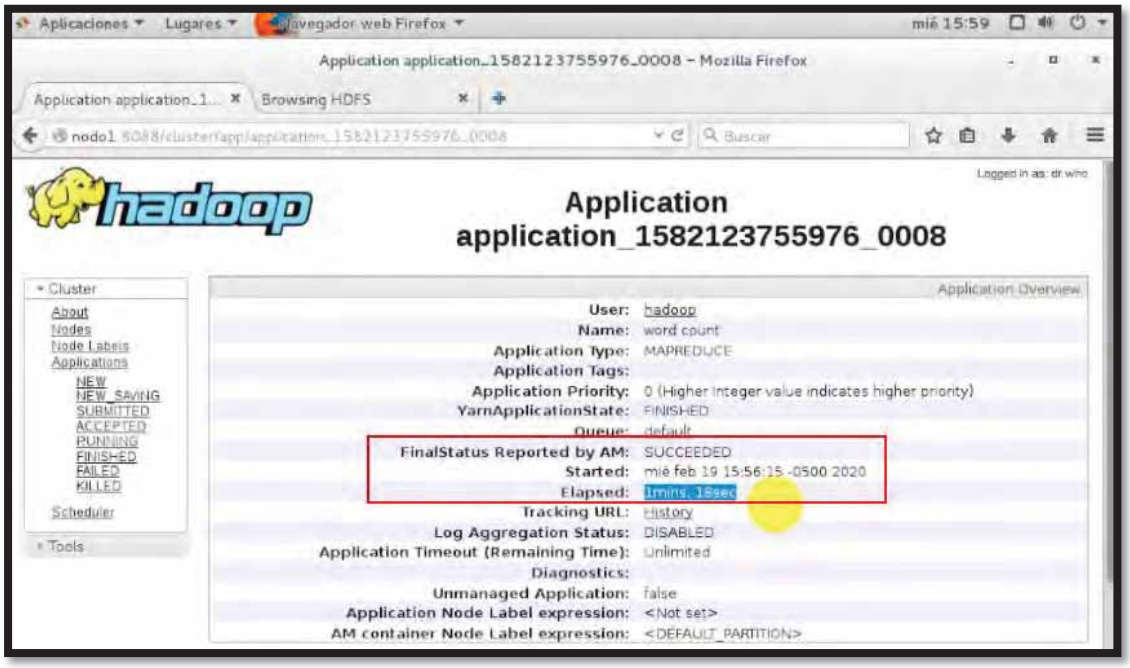


Figura 70. Clúster homogéneo - tiempo de respuesta con 7 nodos: 1 NameNode y 6 DataNodes

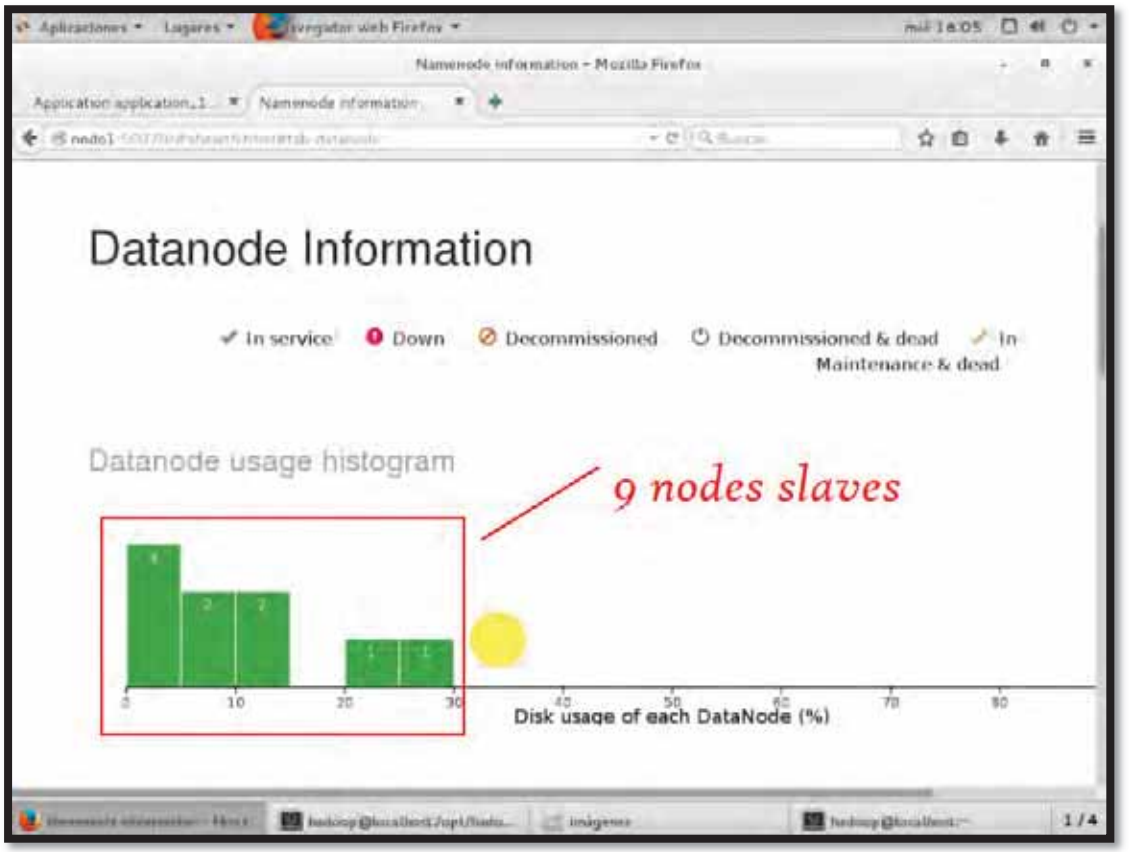


Figura 71. Clúster homogéneo - 9 nodos mostrados en NameNode

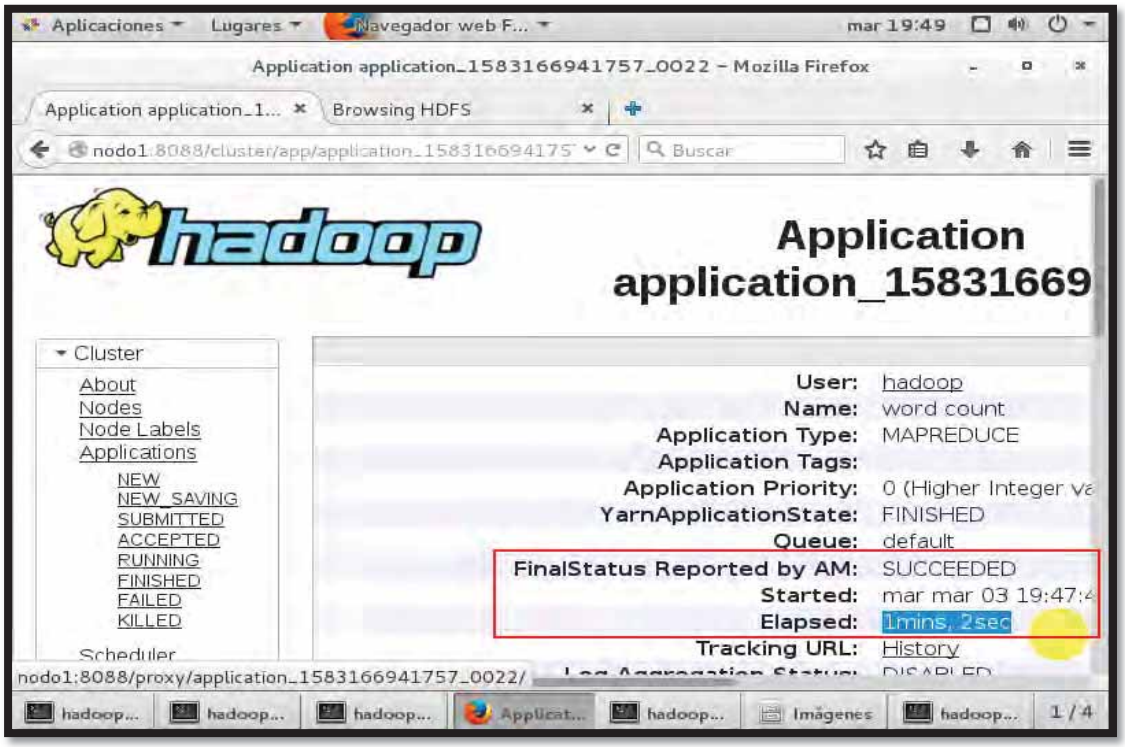


Figura 72. Clúster homogéneo - tiempo de respuesta con 10 nodos: 1 NameNode y 9 DataNodes

4.10. Infraestructura heterogénea de Hadoop.

La infografía muestra 10 nodos, con características diferentes, el primer nodo es maestro y los 9 nodos son esclavos, como referencia la figura 73.

Figura 73. Clúster heterogéneo (fuente propia)

Las características de clúster heterogéneo, se detallan los 10 nodos; Procesadores 5 Intel Xeon, 3 Intel i5 y 2 Intel i7, Disco duro de 10 TB y memoria RAM de 162 GB, como referencia figura 74.

Características PC 1	
Sistema	Lenovo
RAM	16 GB
Disco HDD	1 T
Procesador	Intel(R)Xeon(R) CPU E3-1270 v6 @ 3.8Ghz 3.79GHz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 2	
Sistema	ECS
RAM	8 GB
Disco HDD	1 TB
Procesador	Intel(R) core (TM) i5-3330 CPU @ 3.00Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 3	
Sistema	HP
RAM	10GB
Disco HDD	1 TB
Procesador	Intel(R) core(TM) i5-2500 CPU @ 3.30Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 4	
Sistema	Lenovo
RAM	4 GB
Disco HDD	500GB
Procesador	Intel(R)core(TM) i5-4300U CPU @ 1.90Ghz 3.79GHz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 5	
Sistema	Lenovo
RAM	16 GB
Disco HDD	1 T
Procesador	Intel(R)Xeon(R) CPU E3-1270 v6 @ 3.8Ghz 3.79GHz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características heterogéneas

Características PC 6	
Sistema	Lenovo
RAM	16 GB
Disco HDD	1,8 T
Procesador	Intel(R)core (TM) i7-7700 CPU @ 3.60 Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 7	
Sistema	HP
RAM	30 GB
Disco HDD	1,8 T
Procesador	Intel(R)Xeon(R) W-2123 CPU @ 3.60 Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 8	
Sistema	Lenovo
RAM	16 GB
Disco HDD	1,8 T
Procesador	Intel(R)core (TM) i7-7700 CPU @ 3.60 Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 9	
Sistema	HP
RAM	30 GB
Disco HDD	1,8 T
Procesador	Intel(R)Xeon(R) W-2123 CPU @ 3.60 Ghz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Características PC 10	
Sistema	Lenovo
RAM	16 GB
Disco HDD	1 T
Procesador	Intel(R)Xeon(R) CPU E3-1270 v6 @ 3.8Ghz 3.79GHz
Arquitectura	64 bits
Sistema Operativo	Centos 7
Hadoop	2.9
Java	version JDK 1.8
Dirección IP	Clase C

Figura 74. Características heterogéneas (fuente propia)

Proseguir y recapitular con la instalación desde el sistema operativo CentOS 7, configuración de Hadoop 2.9.2, configuración de parámetros JDK compatible, editar los archivos de core-site.xml, hdfs-site.xml, map-reduce.xml y yarn-site.xml, creación de SSH y todo lo que conlleva **en un solo nodo**, hasta su inicio de sesión en cada nodo heterogéneo.

Configurar de manera distribuida, con configurar los IP clase C (desde 192.168.10.100 hasta 192.168.10.110), comunicación SSH, mediante configuración ssh-keygen y hasta arrancar su inicio de sesión **en 10 nodos heterogéneos**.

Por otro lado, los componentes de hardware de las computadoras o servidores donde el rendimiento se evalúa con la integración de CPU o procesador, memoria RAM, las solicitudes de I/O al disco duro y la transferencia de bytes por medio de la interface de red, indican cómo se está comportando el sistema ante las pruebas de rendimiento ejecutadas, como nodo maestro con características de PC1, como referencia la figura 75.

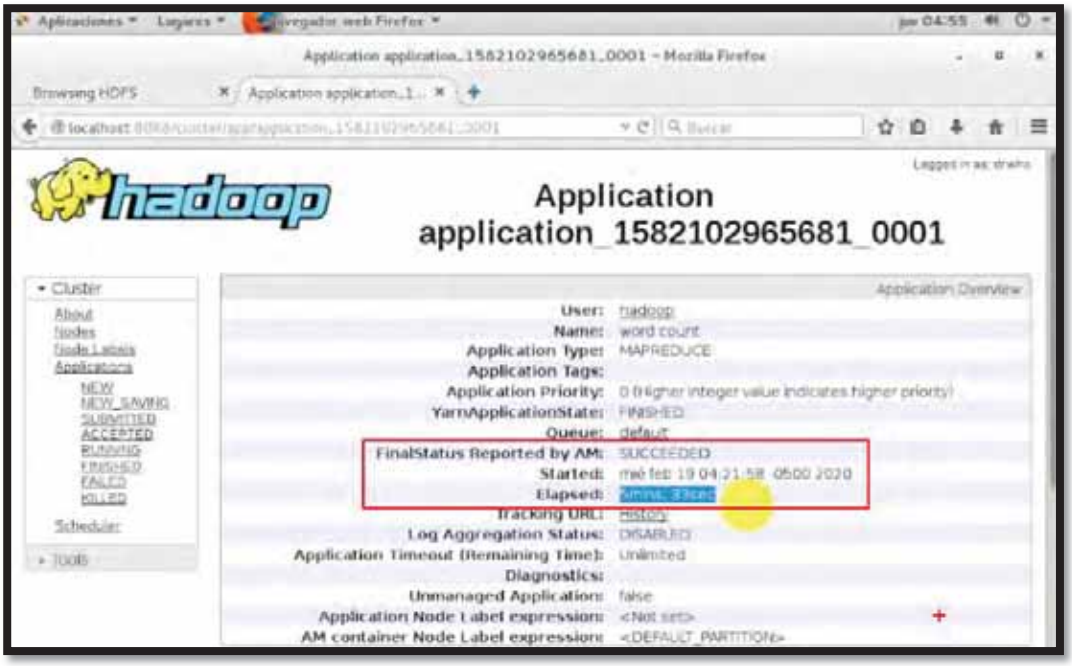


Figura 75. Clúster heterogéneo - tiempo de respuesta con nodomaestro

Adicionando las PC2 y PC3 al clúster heterogéneos y dando resultado la figura 76.

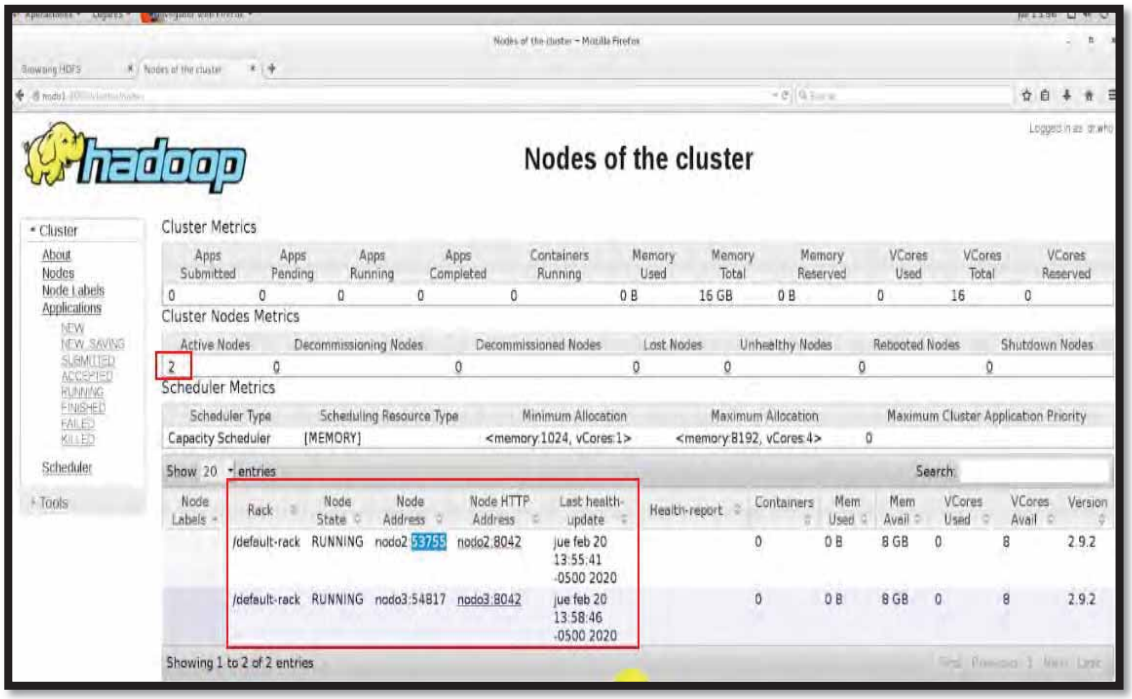


Figura 76. Clúster heterogéneo - 2 nodos mostrados en NameNode

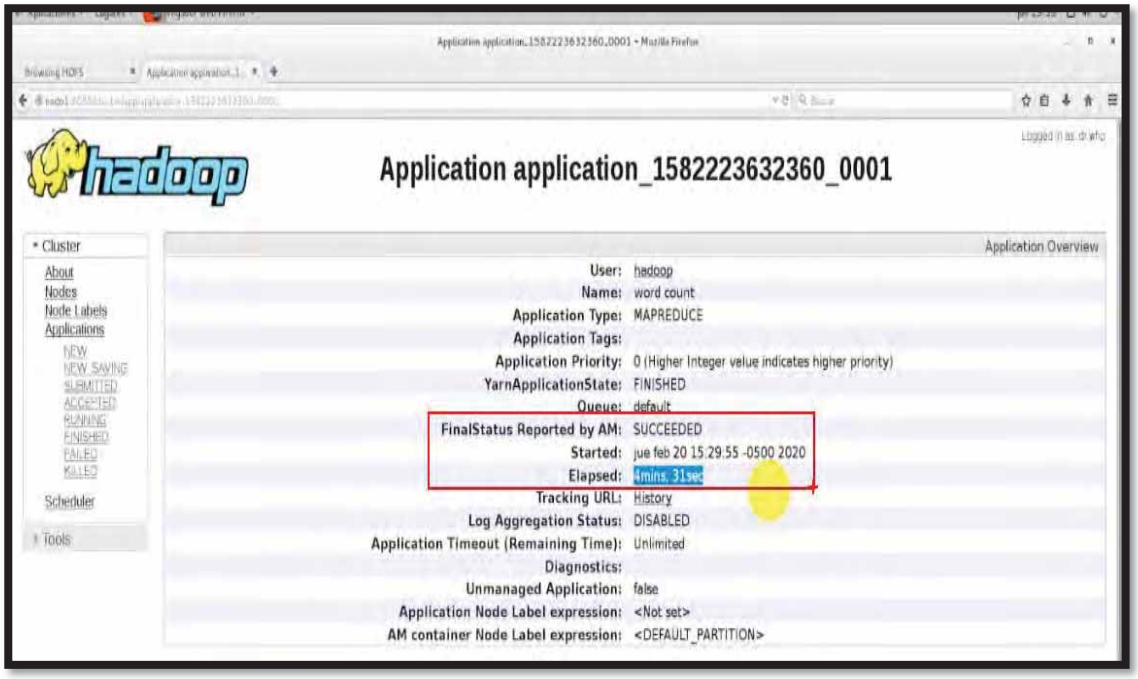


Figura 77. Clúster heterogéneo - tiempo de respuesta con 3 nodos: 1 NameNode y 2 DataNodes

Adicionando las PC4 y PC5 al clúster heterogéneos y dando resultado la figura 78.

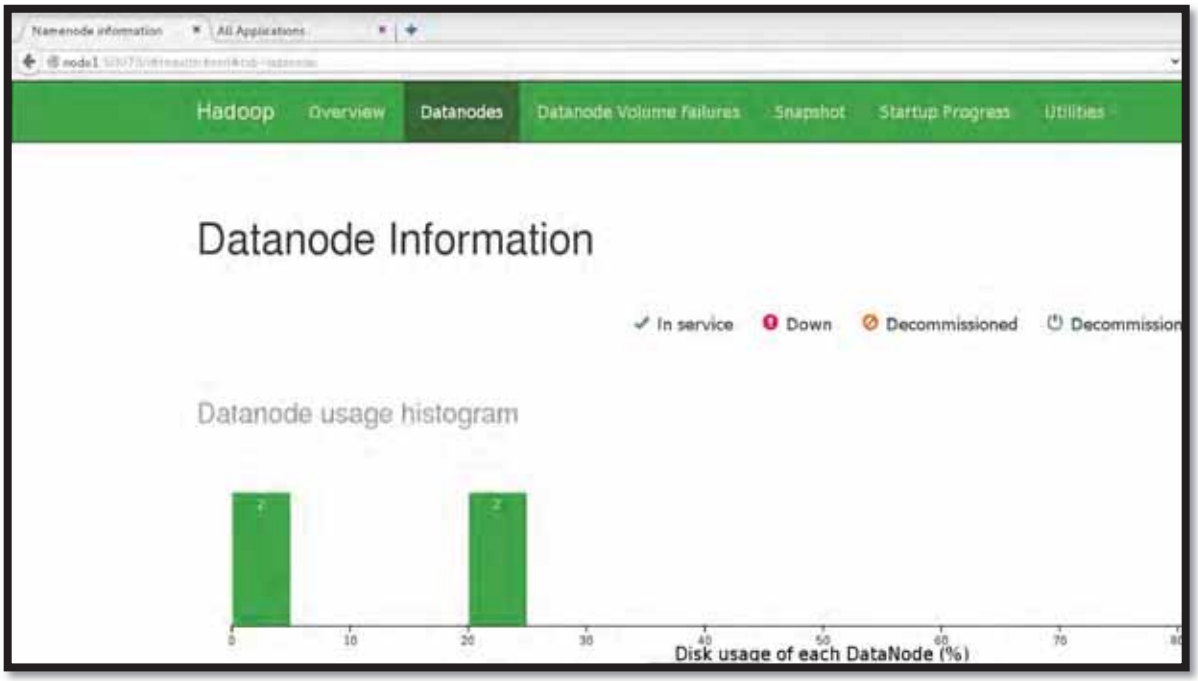


Figura 78. Clúster heterogéneo - 4 nodos mostrados en NameNode

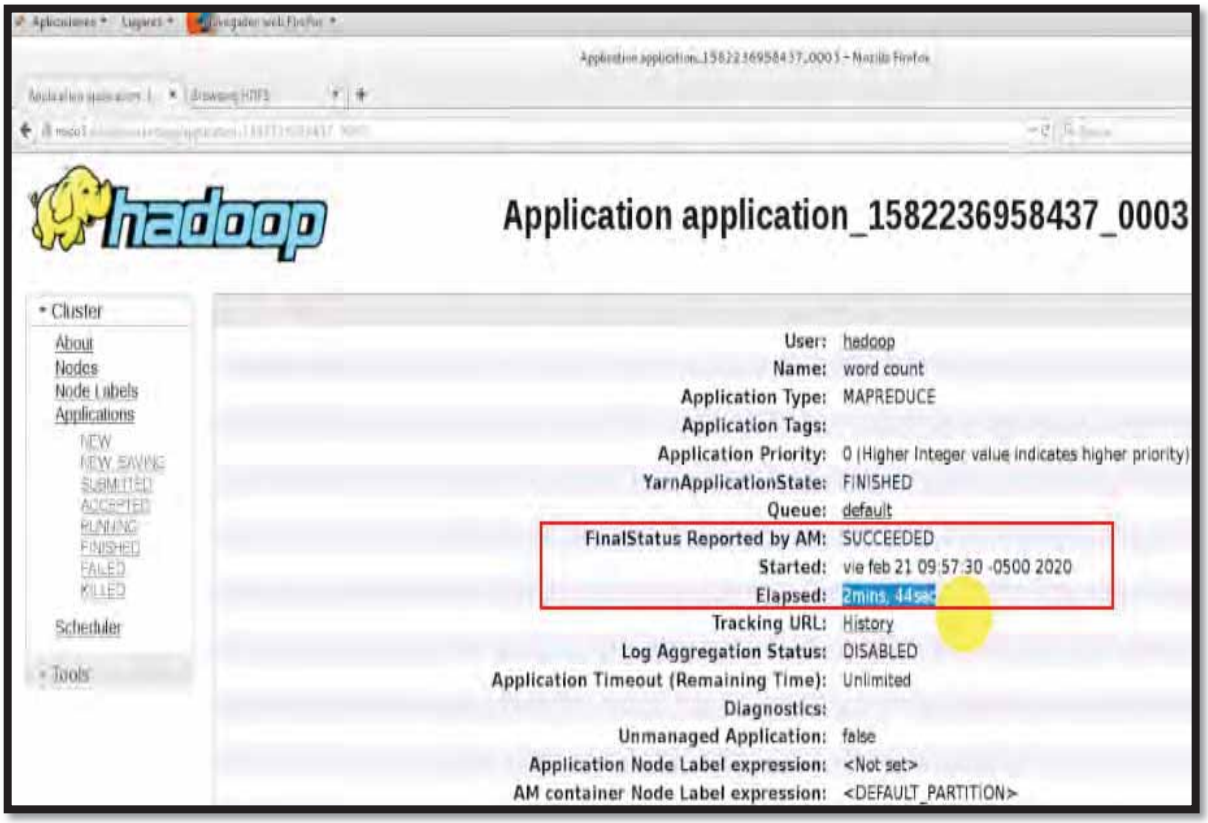


Figura 79. Clúster heterogéneo - tiempo de respuesta con 5 nodos: 1 NameNode y 4 DataNodes

Adicionando las PC6 y PC7 al clúster heterogéneos y dando resultado la figura 80.

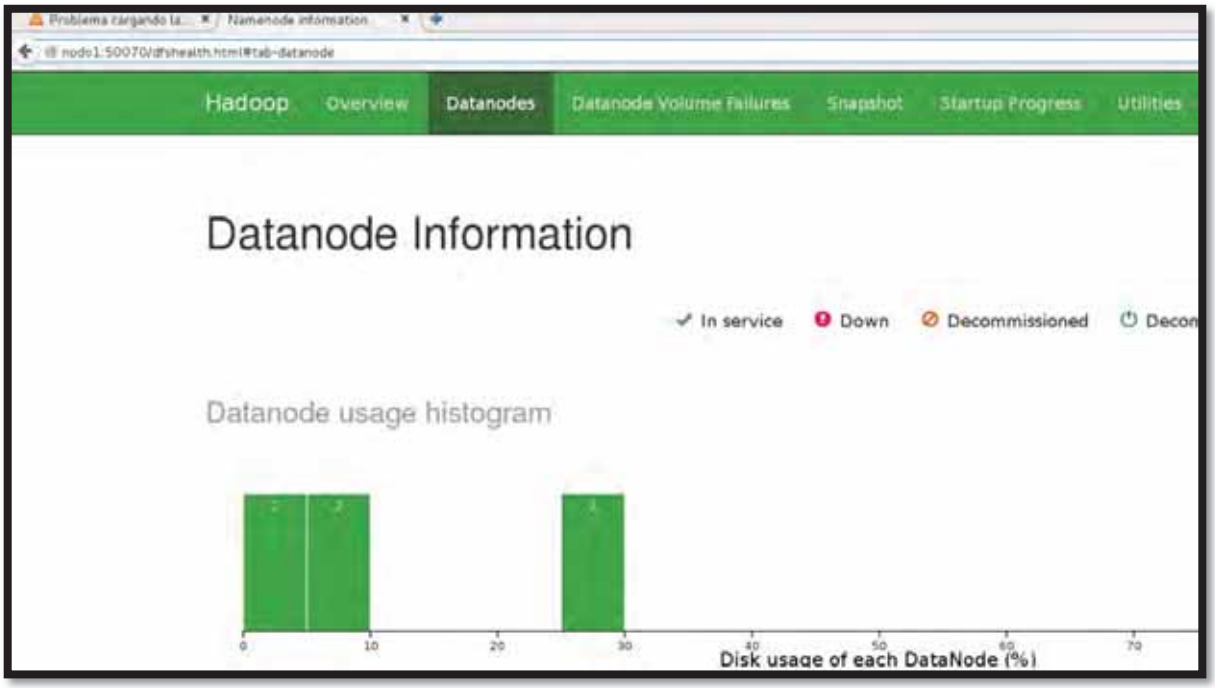


Figura 80. Clúster heterogéneo - 6 nodos mostrados en NameNode, otro enfoque de datanodes

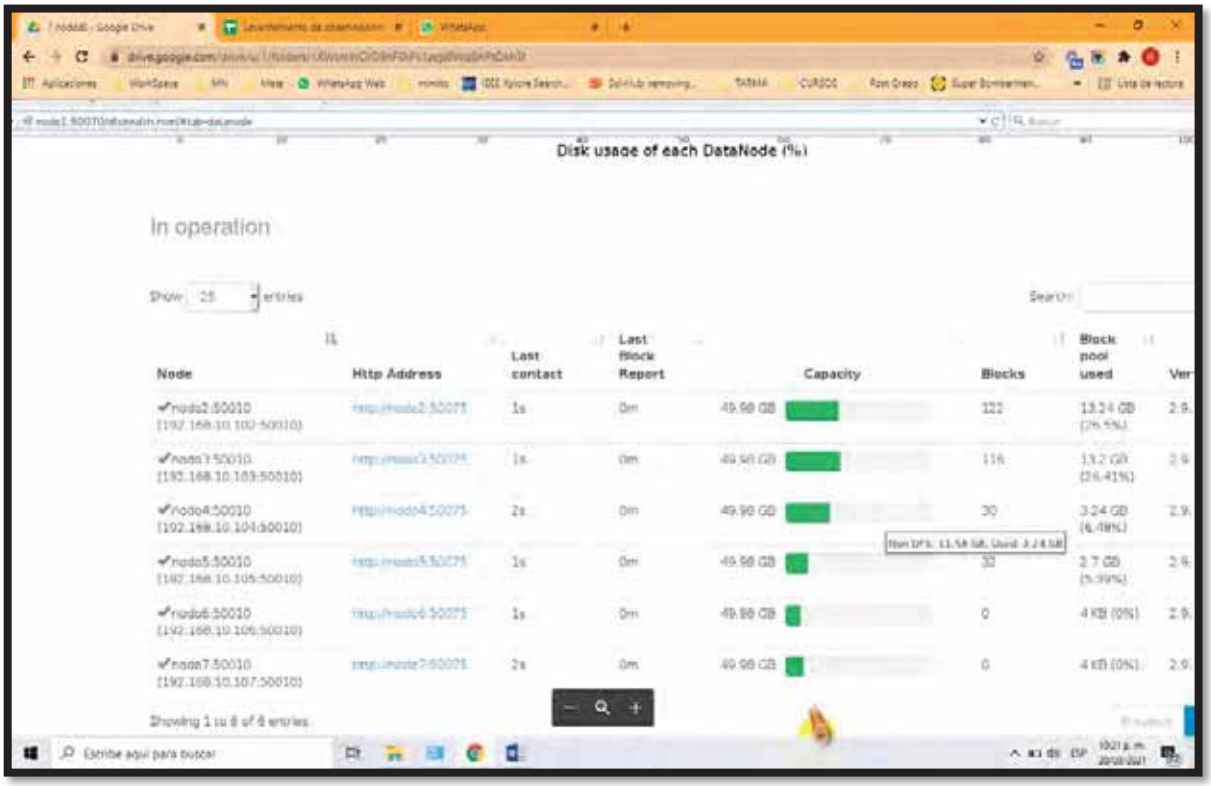


Figura 81. Clúster heterogéneo - 6 nodos mostrados en NameNode

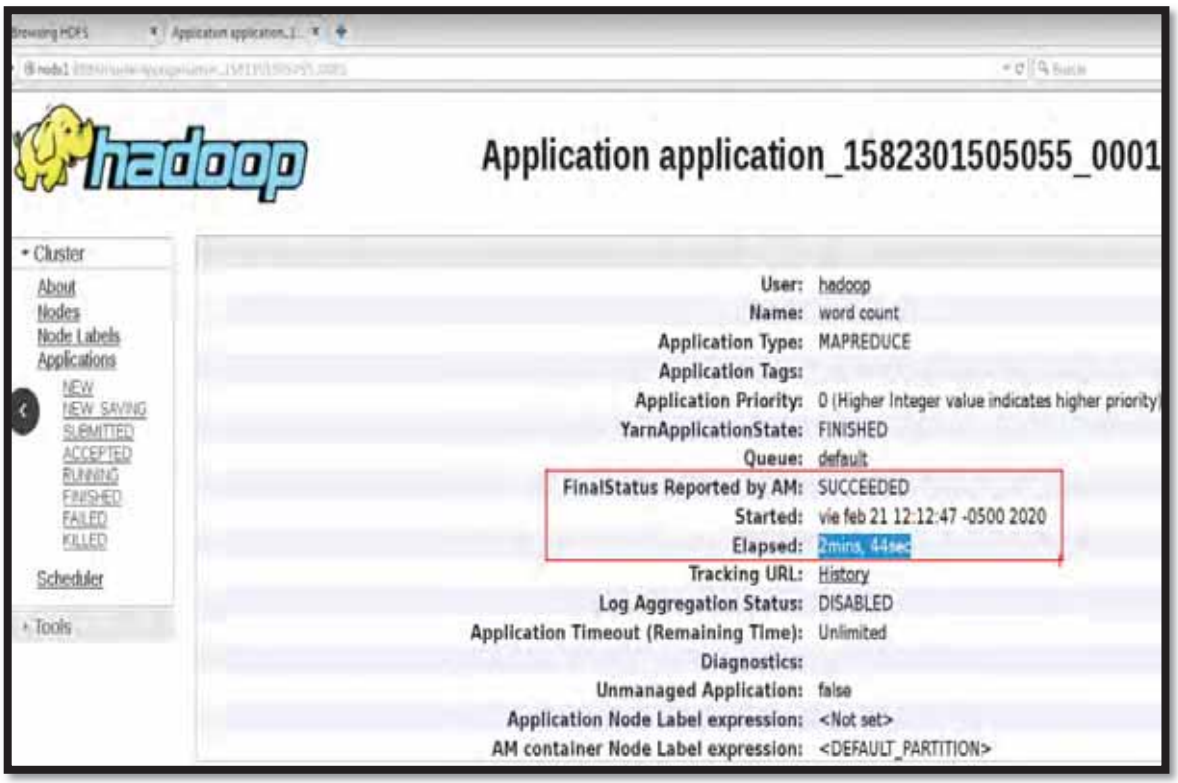


Figura 82. Clúster heterogéneo - tiempo de respuesta con 7 nodos: 1 NameNode y 6 DataNodes

Adicionando las PC8, PC9 y PC10 al clúster heterogéneos y dando resultado la figura 83.

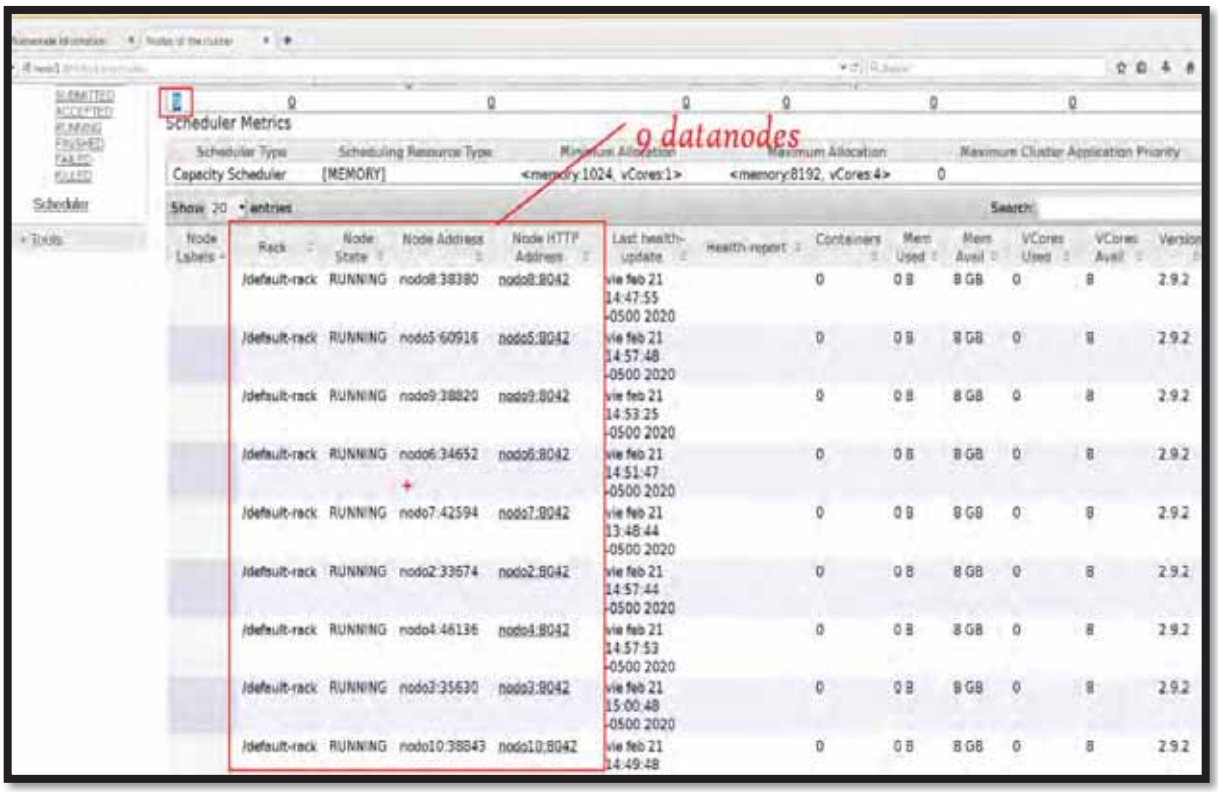


Figura 83. Clúster heterogéneo - 9 nodos mostrados en NameNode

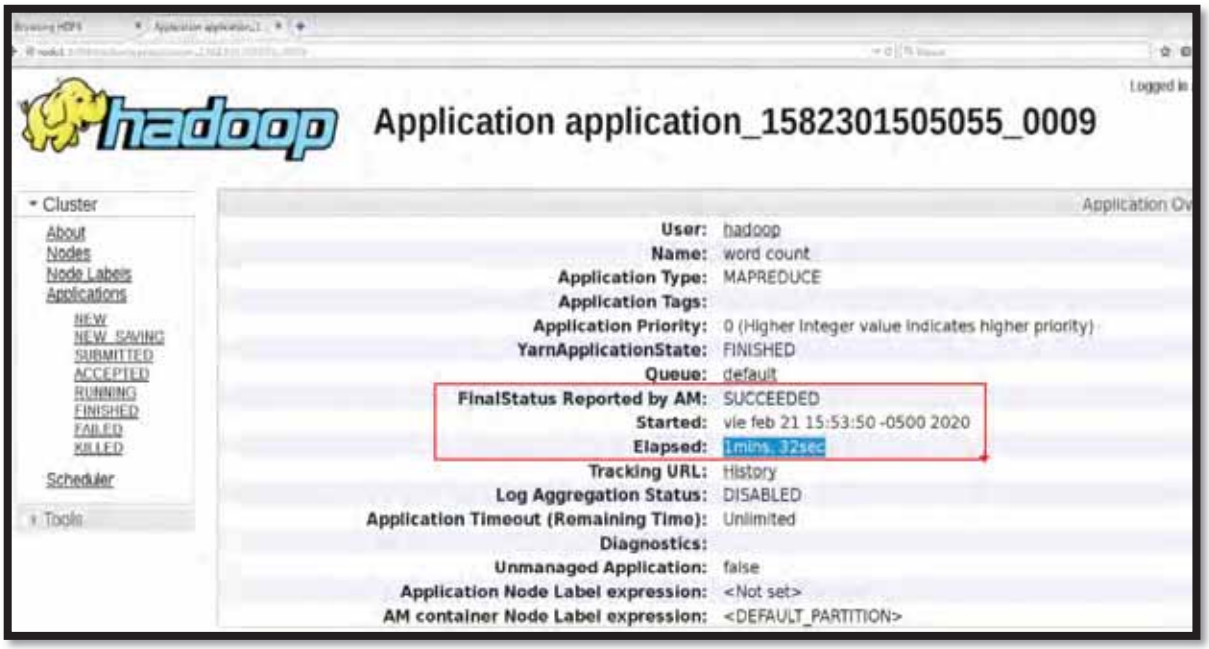


Figura 84. Clúster heterogéneo - tiempo de respuesta con 10 nodos: 1 NameNode y 9 DataNodes

CAPÍTULO V

RESULTADOS Y DISCUSIÓN

5.1.Proceso de rendimiento en clúster homogéneo.

Analizar e interpretar el rendimiento de clúster homogéneo, mediante el algoritmo de modelo de programación de MapReduce en batch secuencialmente y distribución de datos no estructurados (conjunto de libros de www.gutenberg.org) en HDFS separados en bloques predefinidos como 126 MB.

Usar dos variables para poder concretar, estimar el tiempo de rendimiento por cantidad de Computadoras. Usar como variables independientes – cantidad de Computadoras Personales y variables dependientes es el tiempo de rendimiento de procesamiento. ¿Cómo determinar la relación de variables? En consecuencia, iniciaremos con análisis de regresión.

5.1.1. Análisis de regresión.

Se basa en la relación, asociación, entre dos variables, la variable independiente-nodos y la variable dependiente-tiempo de rendimiento, para ello usaremos un aglomerado de libros descargados del repositorio www.gutenberg.org, con una unidad de medida 6.4 GB y así ya detallado en las figuras de clúster homogéneo como sigue:

- Configuramos un PC1 como nodomaestro (NameNode y DataNode)
- Adicionando 2 PC más, consecuencia un clúster de 3 nodos y su tiempo de rendimiento reduce.
- Adicionando 2 PC más, consecuencia un clúster de 5 nodos y su tiempo de rendimiento reduce.

- Adicionando 2 PC más, consecuencia un clúster de 7 nodos y su tiempo de rendimiento reduce.
- Adicionando 3 PC más, consecuencia un clúster de 10 nodos y su tiempo de rendimiento reduce.

Dicha descripción es compendio de las pruebas de las figuras de clúster homogéneo, como indica en la tabla 7.

Tabla 7. Cantidad de PC y tiempo de rendimiento.

Nodos	***Segundos***
1	333.
3	149.
5	89.
7	78.
10	62.

Nota: Tiempo de rendimiento en nodos homogéneos (fuente propia)

Determinando y basándonos en el marco metodológico, como variable independiente a cantidad de nodos en la arquitectura-CNC y variable dependiente, el tiempo de respuesta entre nodos de la arquitectura-TRN.

5.1.2. Diagrama de dispersión.

Entonces el primer paso para determinar si existe una relación entre dos variables es examinar la gráfica de los datos observados (o conocidos). Esta gráfica o dibujo, se llama diagrama de dispersión, como referencia la figura 85.

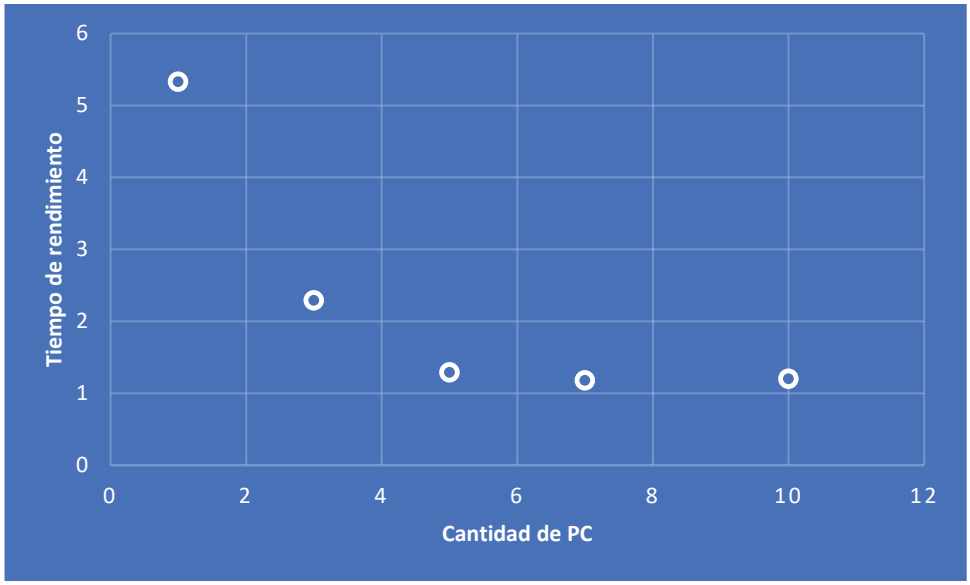


Figura 85. Diagrama de dispersión de características homogéneas (fuente propia)

5.1.3. Tipo de relación de variables.

El tipo de relación es inversa o pendiente negativa a simple vista, pero busquemos mejor estimación de tiempo en la línea que determina tal relación de variables.

Relación inversa entre x e y, que representan a cantidad de nodos y tiempo de rendimiento respectivamente en este estudio, la variable dependiente disminuye al aumentar la variable independiente, como muestra la figura 86.

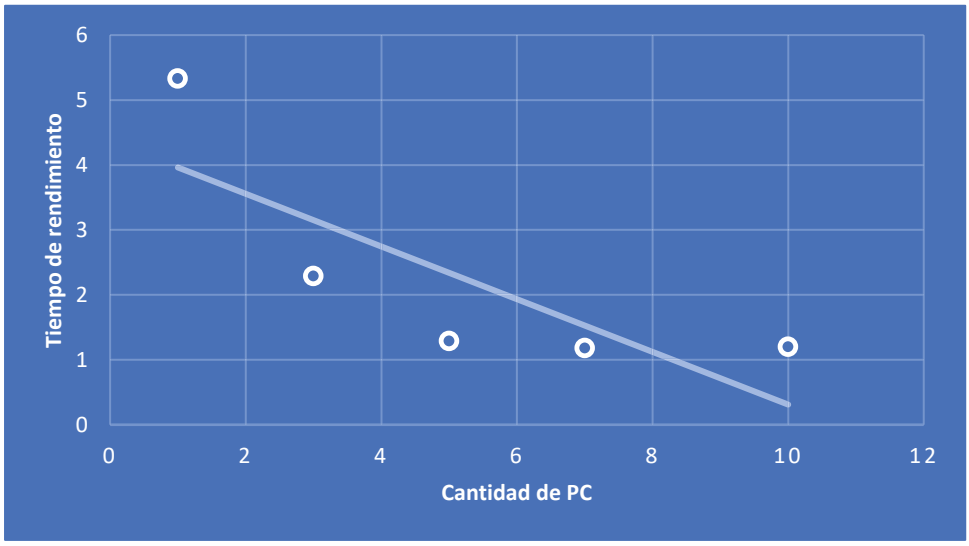


Figura 86. Tipo de relación de variables (fuente propia)

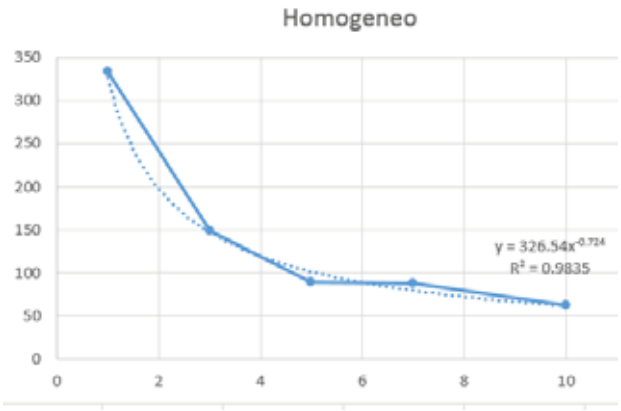


Figura 87.- Línea de tipo potencial.

Acotando con mayor precisión a nuestro análisis, con esta figura 87, para conseguir una línea de tendencia de tipo potencial, que se aproxima mucho más a la definición “Que, a mayor cantidad de nodos, menor el tiempo de ejecución”.

5.1.4. Estimación mediante la recta de regresión.

Calcular la línea de regresión de manera más precisa, usando la ecuación que relaciona las dos variables matemáticamente. La ecuación para una línea recta donde la variable dependiente "y" está determinada por la variable independiente "x" es, como referencia figura 88. (Levin, 2004).

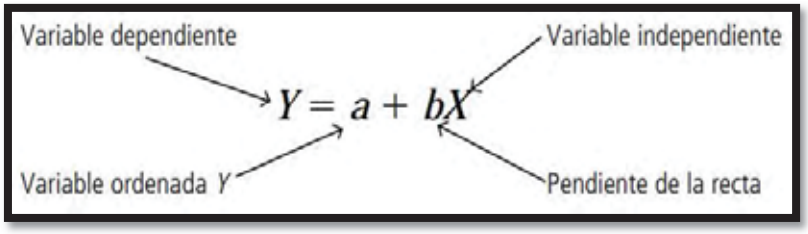


Figura 88. Representa ecuación para la línea recta

Entonces determinar con precisión el diagrama de dispersión y el tipo de relación de dichas variables dependientes (cantidad de nodos) e independiente (tiempo de rendimiento) es de relación inversa. Tenemos la ecuación que determina con

precisión: y = -0.4057x +4.3676, donde la pendiente negativo de -0.4057 y la ordenada "y" es 4.3676, como referencia la figura 89.

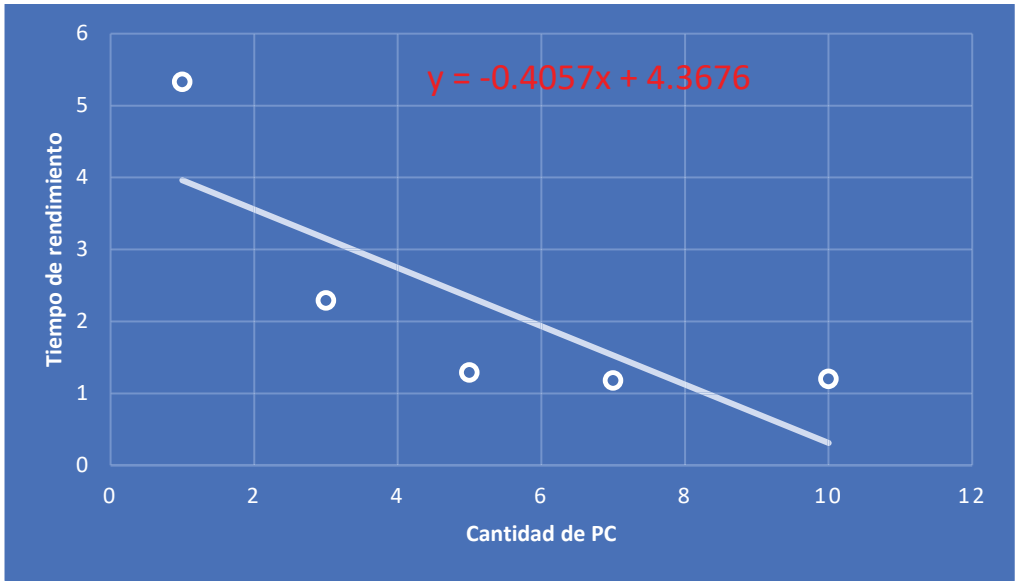


Figura 89. La ecuación que representa la estimación (fuente propia)

5.1.5. Método de mínimos cuadrados

¿Cómo podemos "ajustar" una recta matemáticamente si ninguno de los puntos está sobre ella? Mediante el ajuste matemático de una recta de regresión.

La línea tendrá un "buen ajuste" si minimiza el error entre los puntos estimados en la recta y los puntos observados reales que se utilizaron para trazarla.

Ahora encontrar matemáticamente la recta de mínimos cuadrados que mejor se ajusta, con la nueva representación de "ye gorra", como la figura 90. (Levin, 2004)

$$\hat{Y} = a + bX$$

Figura 90. Línea de estimación

Uso del error total para determinar el mejor ajuste. Parece razonable que mientras más lejos esté un punto de la línea de estimación, más serio es el error.

Preferiríamos tener varios errores absolutos pequeños que uno grande, como vimos en el ejemplo anterior. En efecto, deseamos encontrar una forma de "penalizar" errores absolutos grandes, para poder evitarlos. Podemos lograr esto si elevamos al cuadrado los errores individuales antes de sumarlos. Los cuadrados de cada término logran dos objetivos:

1. Magnifica, o penaliza, los errores más grandes.
2. Cancela el efecto de los valores positivos y negativos (un error negativo al cuadrado sigue siendo positivo).

Como estamos buscando la línea de estimación que minimiza la suma de los cuadrados de los errores, a esto le llamamos método de mínimos cuadrados.

Los estadísticos han desarrollado dos ecuaciones que podemos utilizar para encontrar la pendiente y la ordenada Y de la recta de regresión de mejor ajuste. La primera fórmula calcula la pendiente, referencia figura 91. (Levin, 2004)

$$b = \frac{\Sigma XY - n\overline{X}\overline{Y}}{\Sigma X^2 - n\overline{X}^2}$$

donde,

- b = pendiente de la línea de estimación de mejor ajuste
- X = valores de la variable independiente
- Y = valores de la variable dependiente
- $\overline{X}$ = media de los valores de la variable independiente
- $\overline{Y}$ = media de los valores de la variable dependiente
- n = número de puntos

Figura 91. Pendiente de la recta de regresión de mejor ajuste.

La segunda fórmula calcula la ordenada Y de la recta, referencia figura 92. (Levin, 2004)

$$a = \overline{Y} - b\overline{X}$$

donde,

- a = ordenada Y
- b = pendiente de la ecuación

Figura 92. Ordenada Y de la recta de regresión de mejor ajuste

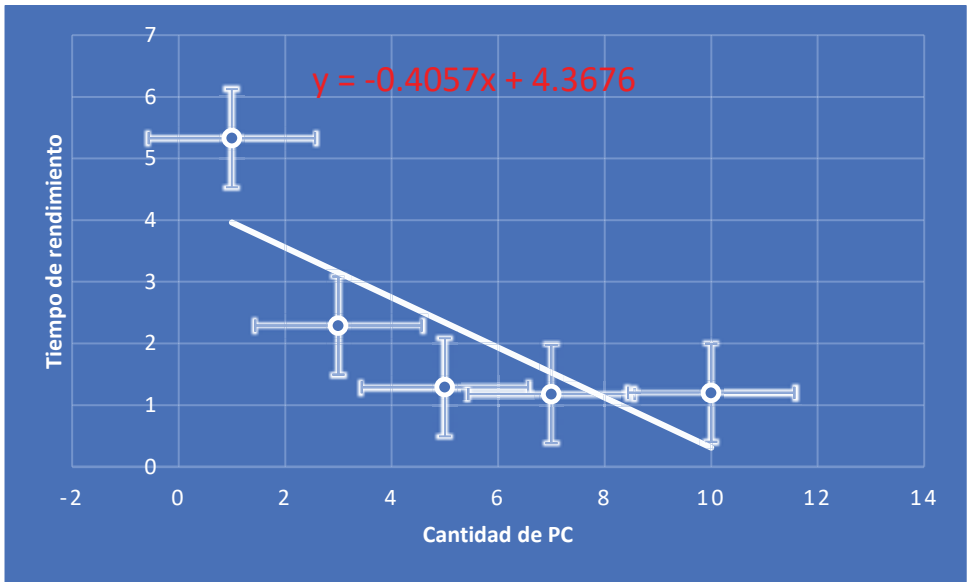


Figura 93. Ordenada Y de la recta de regresión de mejor ajuste (fuente propia)

En consecuencia, a la figura 93 y fórmulas de mejor ajuste de la recta de regresión, el error de la pendiente es - 0.4057 $\pm$ 0.1777, como el error de la ordenada al origen es 4.3676 $\pm$ 1.0782.

5.1.6. Error estándar de estimación

Uso de un método abreviado para calcular el error estándar de la estimación, como referencia figura 94. (Levin, 2004)

$$s_e = \sqrt{\frac{\Sigma Y^2 - a\Sigma Y - b\Sigma XY}{n-2}}$$

Figura 94. Método abreviado de error estándar de la estimación.

La resolución del método abreviado S_e es 1.2416 y como referencia de ubicar en la figura 95. (Levin, 2004, pág. 529)

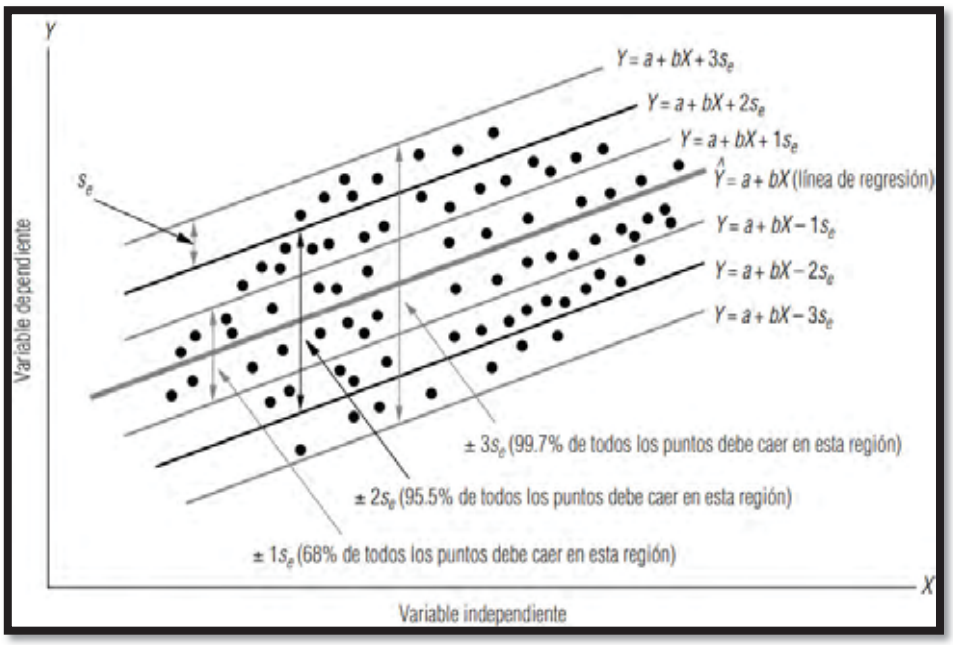


Figura 95. límites alrededor de la línea de regresión de ±1Se, ±2Se y ±3Se

5.1.7. Análisis de correlación

Es la herramienta estadística que podemos usar para describir el grado en el que una variable está linealmente relacionada con otra.

Hay dos medidas que describen la correlación entre dos variables: El coeficiente de determinación y el coeficiente de correlación.

Coeficiente de determinación, como referencia la figura 96 y coeficiente de correlación, referencia figura 97. (Levin, 2004)

$$r^2 = \frac{a\Sigma Y + b\Sigma XY - n\bar{Y}^2}{\Sigma Y^2 - n\bar{Y}^2}$$

Figura 96. Método abreviado, coeficiente de determinación.

Reemplazando en la fórmula r^2 es 0.6346.

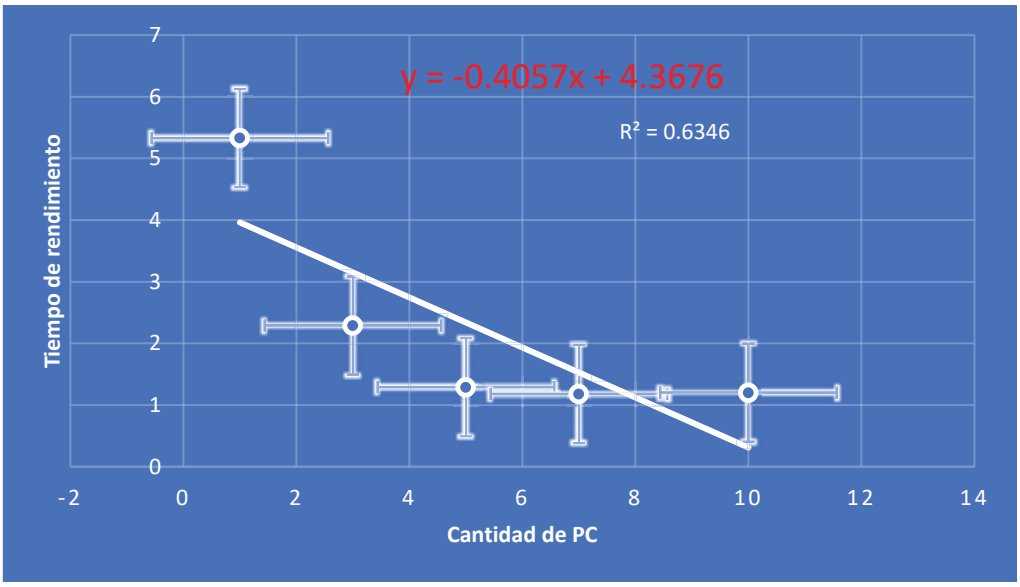


Figura 97. Estimar mediante la recta de regresión (fuente propia)

Coeficiente de correlación, cuando la pendiente de la ecuación de estimación es positiva, r es la raíz cuadrada positiva, pero si es negativa, r es la raíz cuadrada negativa. Entonces, el signo de r indica la dirección de la relación entre las dos variables X y Y, como referencia la figura 98. (Levin, 2004)

$$r = \sqrt{r^2}$$

Figura 98. Coeficiente de correlación.

Dando resultado de r es - 0.7966.

5.2. Proceso de rendimiento en clúster heterogéneo.

Análisis e interpretación en PC o nodos con características heterogéneas.

5.2.1. Análisis de regresión

Utilizar un aglomerado de unidad de medida 6.4 GB. ya detallado en las figuras de clúster heterogéneo detallado de acuerdo a la figura 74, como sigue:

- Configuramos un PC1 como nodomaestro (NameNode y DataNode)

- Adicionando PC 2 y PC 3 detallado de acuerdo a la figura 74, consecuencia un clúster de 3 nodos y su tiempo de rendimiento reduce.
- Adicionando PC 4 y PC 5 detallado de acuerdo a la figura 74, consecuencia un clúster de 5 nodos y su tiempo de rendimiento reduce.
- Adicionando PC 6 y PC7 detallado de acuerdo a la figura 74, consecuencia un clúster de 7 nodos y su tiempo de rendimiento se mantiene.
- Adicionando PC 8, PC 9 y PC 10 detallado de acuerdo a la figura 74, consecuencia un clúster de 10 nodos y su tiempo de rendimiento reduce.

Dicha descripción es compendio de las pruebas de las figuras de clúster heterogéneo, como indica las variables se muestran en la tabla 8.

Tabla 8. Cantidad de PC y tiempo de rendimiento.

Nodos	***Segundos***
1	333
3	271
5	164
7	164
10	91

Nota: El Test de pruebas en nodos heterogéneos. (fuente propia)

Ya está determinando como variable independiente a cantidad de nodos y variable dependiente el tiempo de rendimiento.

5.2.2. Diagrama de dispersión

En clúster heterogéneo, como referencia la figura 99.

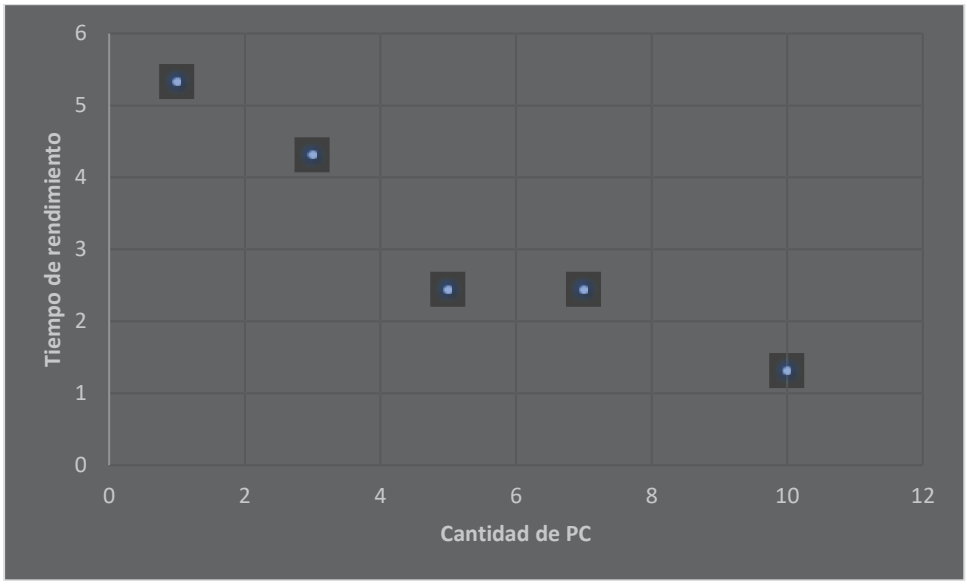


Figura 99. Dispersión heterogénea (fuente propia)

5.2.3. Tipo de relación de variables

Determinar pendiente positiva o pendiente negativa, como referencia la figura 100.

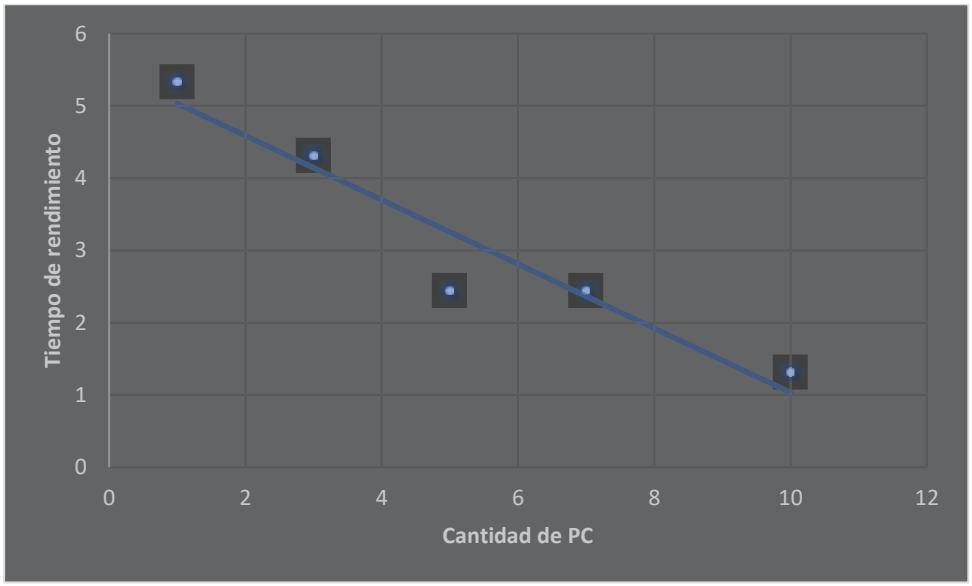


Figura 100. Tipo de relación de variables (fuente propia)

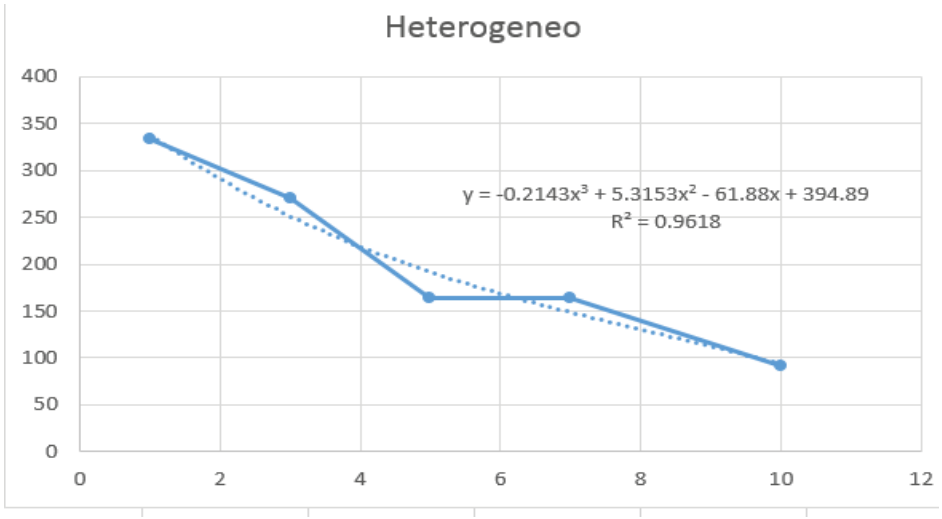


Figura 101.- Línea de tendencia polinómica, de clúster heterogéneo

Adicionando con mayor precisión a nuestro análisis de clúster heterogéneo, con la figura 101, se consigue una línea de tendencia polinómica de orden 3, que se aproxima mucho más a la definición "Que, a mayor cantidad de nodos, menor el tiempo de ejecución".

5.2.4. Estimación mediante la recta de regresión.

Hallar la línea de regresión de manera más precisa, como referencia figura 103.

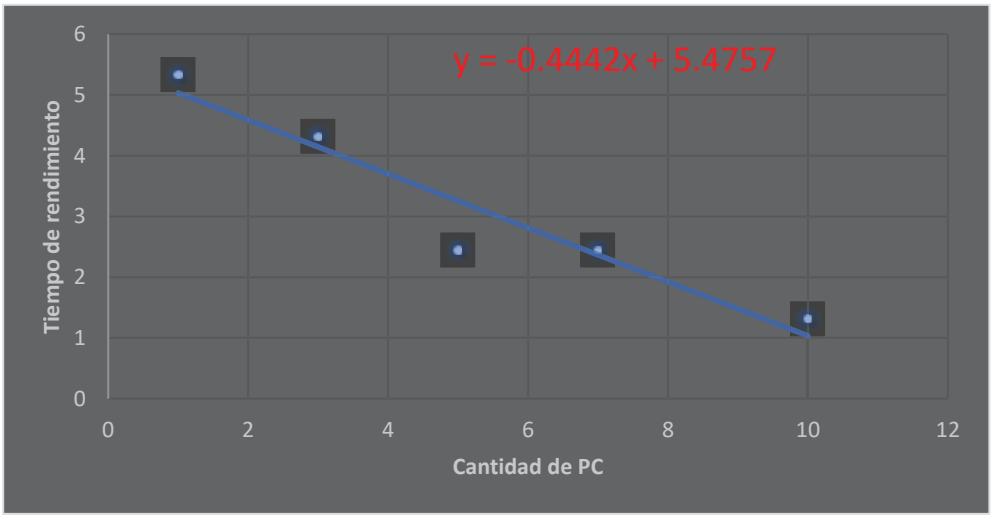


Figura 102. Estimar mediante la recta de regresión (fuente propia)

5.2.5. Método de mínimos cuadrados

Minimizar el error entre los puntos estimados en la recta y los puntos observados y obtener “buen ajuste” de la línea de regresión, como referencia la figura 104.

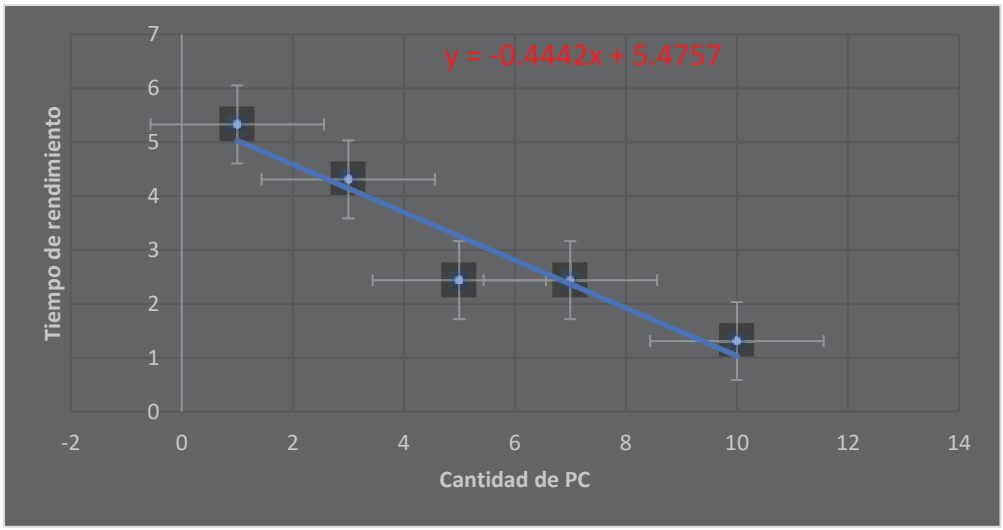


Figura 103. Ordenada Y de la recta de regresión de mejor ajuste (fuente propia)

En consecuencia, al gráfico y fórmulas de mejor ajuste de la recta de regresión, el error de la pendiente es -0.4442 $\pm$ 0.0768, como el error de la ordenada al origen es 5.4758 $\pm$ 0.4656.

5.2.6. Error estándar de estimación

Resultado de la formula S_e es 0.5362 y como referencia la figura 105. (Levin, 2004)

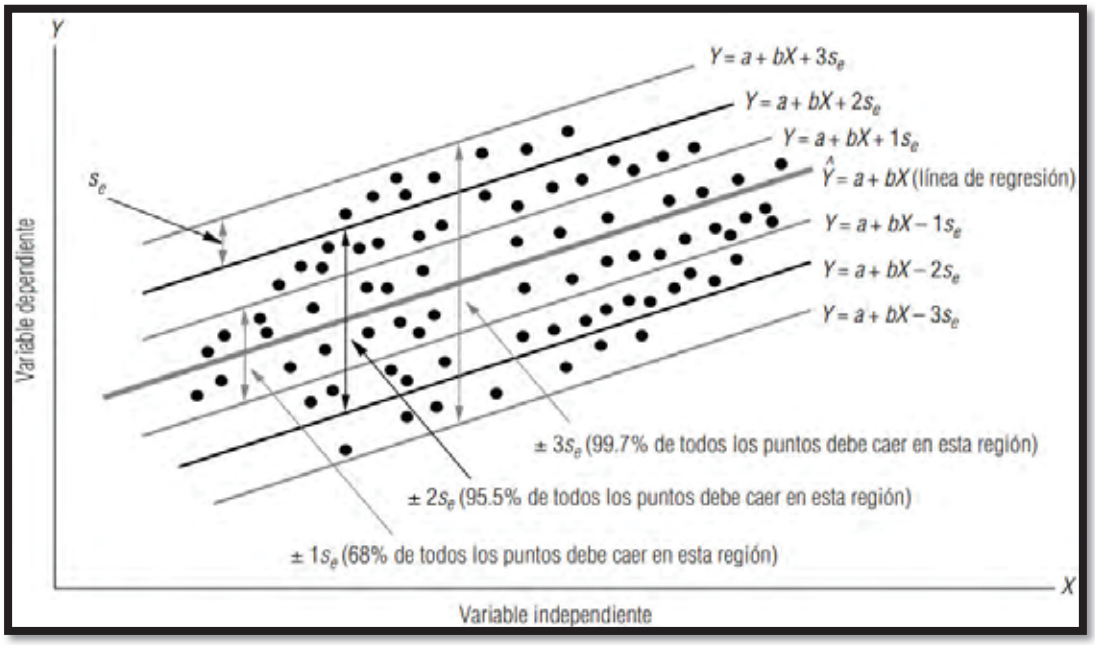


Figura 104. Límites alrededor de la línea de regresión de $\pm 1S_e$, $\pm 2S_e$ y $\pm 3S_e$

5.2.7. Análisis de correlación

Determinar el grado que una variable está linealmente relacionada con otra, Primero mediante coeficiente de determinación r^2 es 0.9178, segundo coeficiente de correlación, resultado de r como - 0.9580, como referencia la figura 106.

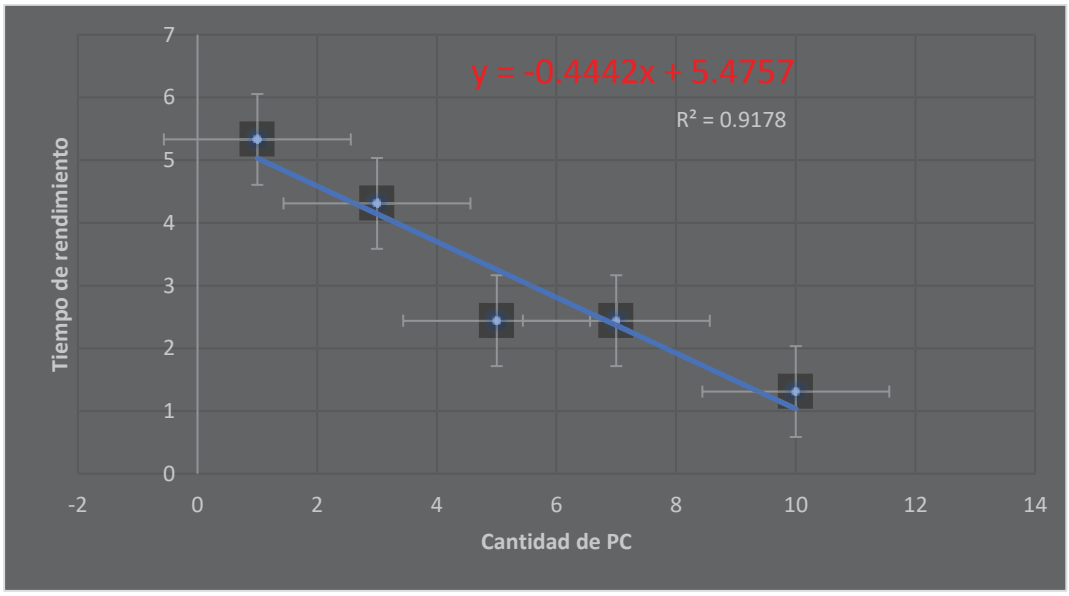


Figura 105. Estimar mediante la recta de regresión (fuente propia).

5.3.Interpretación y discusión.

5.3.1. Descripción sobresaliente de características y conclusiones.

A las primeras 3V, son características de Gartner (2011), que conlleva a volumen, velocidad y variedad. Se adiciona Ashish Juneja (2019) la **veracidad**: Trata de la exactitud, veracidad de los datos, y si es auténtico. Y por último **valor**: Trata de la valía extraída a partir de varios datos brutos disponibles. Se **optimiza** la definición de Big Data.

5.3.2. Interpretar hardware homogéneo vs heterogéneo

Hardware integrado total de cada clúster; Memoria de 162 GB de RAM de clúster heterogéneo es mayor en 2GB RAM que clúster homogéneo.

Disco duro de10 TB HDD de clúster homogéneo es mayor en 0.9 TB HDD de clúster heterogéneo.

10 procesadores Intel Xeon integran clúster homogéneo mientras el clúster heterogéneo está integrado por 5 procesadores Intel Xeon, 3 procesadores Intel i5 y 2 procesadores Intel i7, referencia figura 8107.

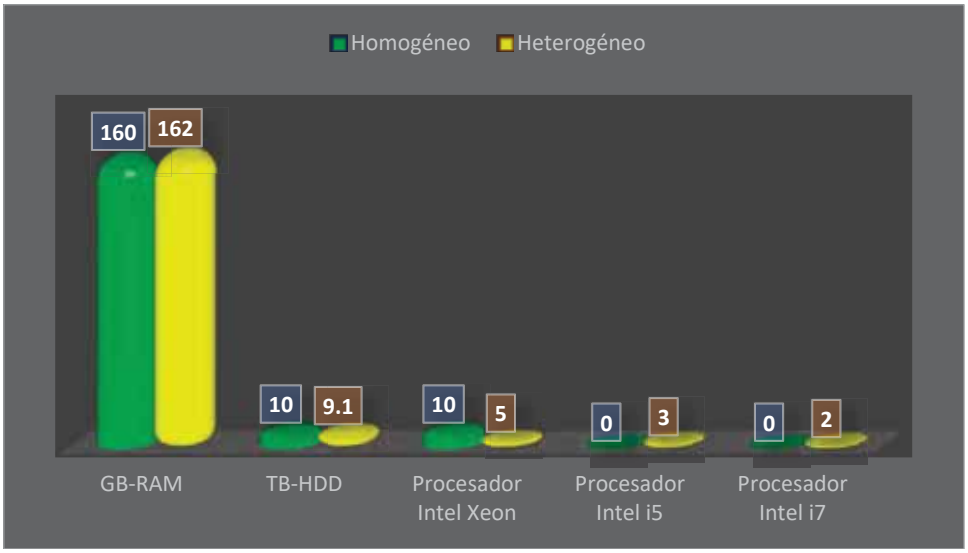


Figura 106. Hardware integrado en homogéneo y heterogéneo (fuente propia).

5.3.3. Interpretar tiempo de rendimiento

Las barras rojas indican la cantidad de PC o nodos, barras celestes y negras indican tiempo de rendimiento en clúster homogéneo y clúster heterogéneo respectivamente.

Resultado es mientras más cantidad de PC o nodos, menos el tiempo de rendimiento, como referencia la figura 108.

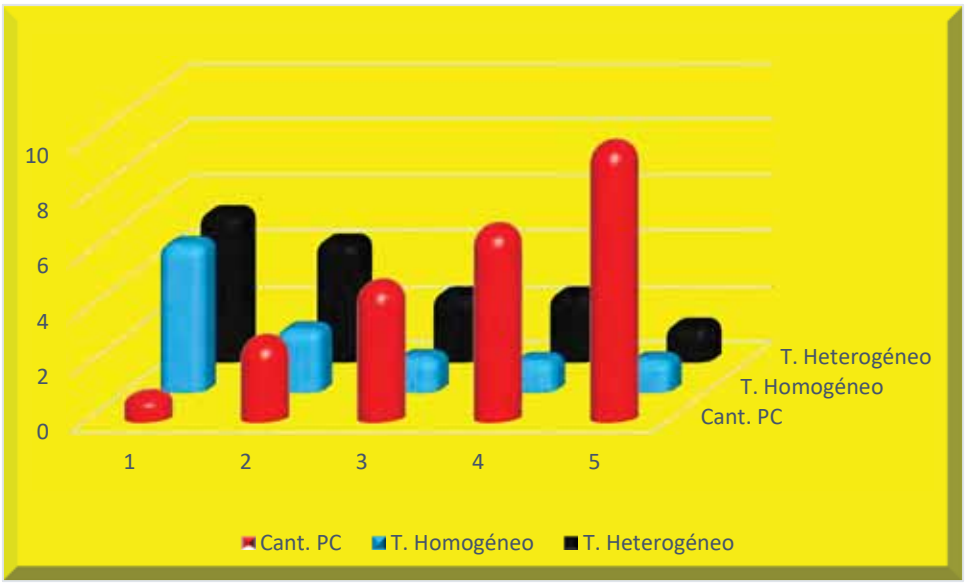


Figura 107. Contraste de estimación mediante recta de regresión (fuente propia)

5.3.4. Interpretar y contrastar la estimación mediante recta de regresión

Determina la naturaleza de dispersión más compacta en clúster heterogéneo y menos compacto en clúster homogéneo, tipo de pendiente es negativo o inversa en ambos clústeres. La recta de regresión heterogénea es más compacta matemáticamente con Y= - 0.4442 + 5.4557, mientras en clúster homogénea es Y= - 0.4057 + 4.3676, como referencia tabla 9 y figura 109 respectivamente.

Tabla 9. Contraste de recta de regresión.

Cantidad de PC	***Homogénea***	***Heterogénea***

Dispersión	Menos compacto.	Más compacto.
Pendiente	Negativo o inversa.	Negativo o inversa.
Error estimado	máximo.	mínimo.

Nota: El Test de pruebas en nodos homogéneos. (fuente propia)

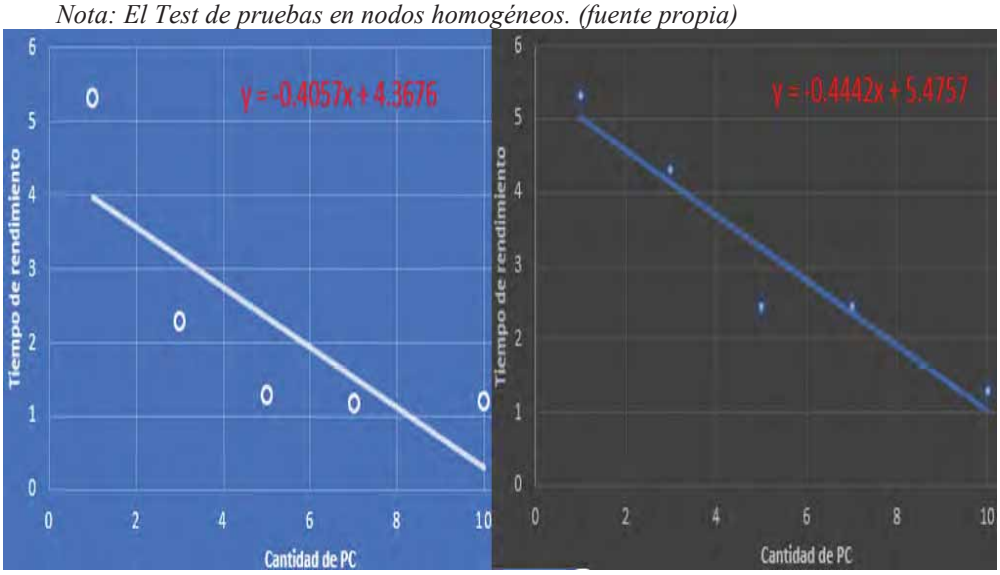


Figura 108. Contraste, estimación mediante recta de regresión (fuente propia).

5.3.5. Contraste de mínimos cuadrados.

El rango límite de error estándar de estimación S_e homogéneo es 1.2416, con su equivalente de 124% de límite de rango $\pm$ 2 S_e.

Mientras el rango límite de ubicación más compacto es de clúster heterogéneo, numéricamente el error estándar de estimación S_e es 0.5362, con su equivalente 54% de caer en esta región $\pm$ 1 S_e, referencia figura 110. (Levin, 2004)

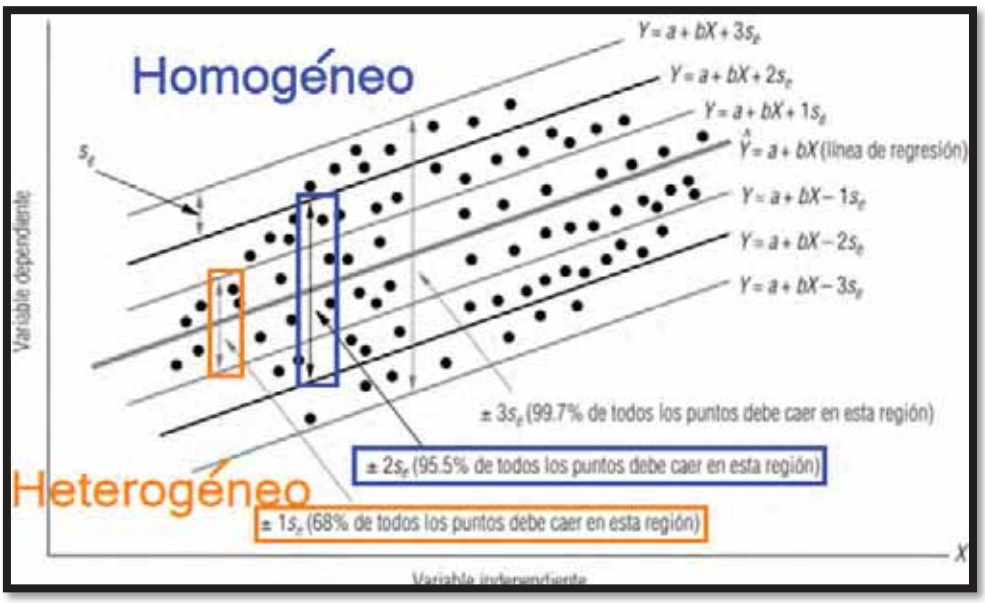


Figura 109. Límites alrededor de la línea de regresión de ±1Se, ±2Se y ±3Se (Levin, 2004)

5.3.6. Contraste, análisis de correlación.

En clúster homogénea, en base a su coeficiente de determinación, damos resultado el coeficiente de correlación r es 0.7966, que significa el 80% de datos se relacionan entre sí y su dirección depende del signo de su pendiente.

En clúster heterogéneo, en base a su coeficiente de determinación podemos hallar el coeficiente de correlación como r es 0.9580, que su equivalente significa en porcentaje 96% de los datos se relacionan entre sí y su dirección depende del signo de su pendiente.

El coeficiente de correlación heterogénea, estima que el 96% de la cantidad de PC o nodos están más estrechamente relacionados con su tiempo de rendimiento, que clúster homogéneo, referencia figura 111.

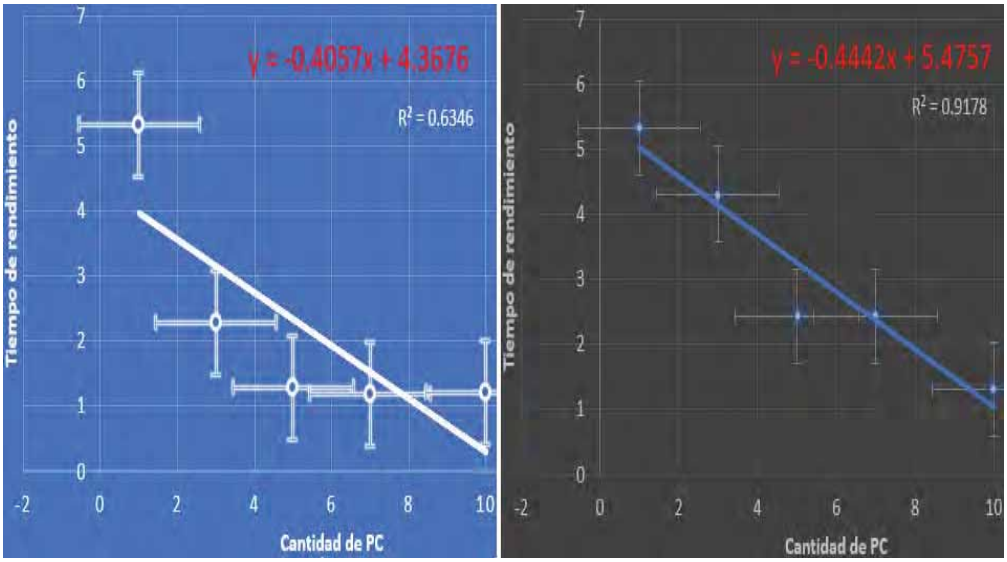


Figura 110. Contraste, coeficiente de determinación (fuente propia).

5.3.7. Rendimiento en clúster homogénea.

Para definir mejor estimación en clúster homogénea, se hacen pruebas de rendimiento con 12.8 GB y 6.4 GB. Las barras rojas indican en el grafico la cantidad de PC o nodos ascienden de menos a más, en consecuencia, las barras amarillas y blancas son su tiempo de rendimiento de 12.8 GB y 6.4 GB respectivamente, descienden de más a menos, referencia figura 112.

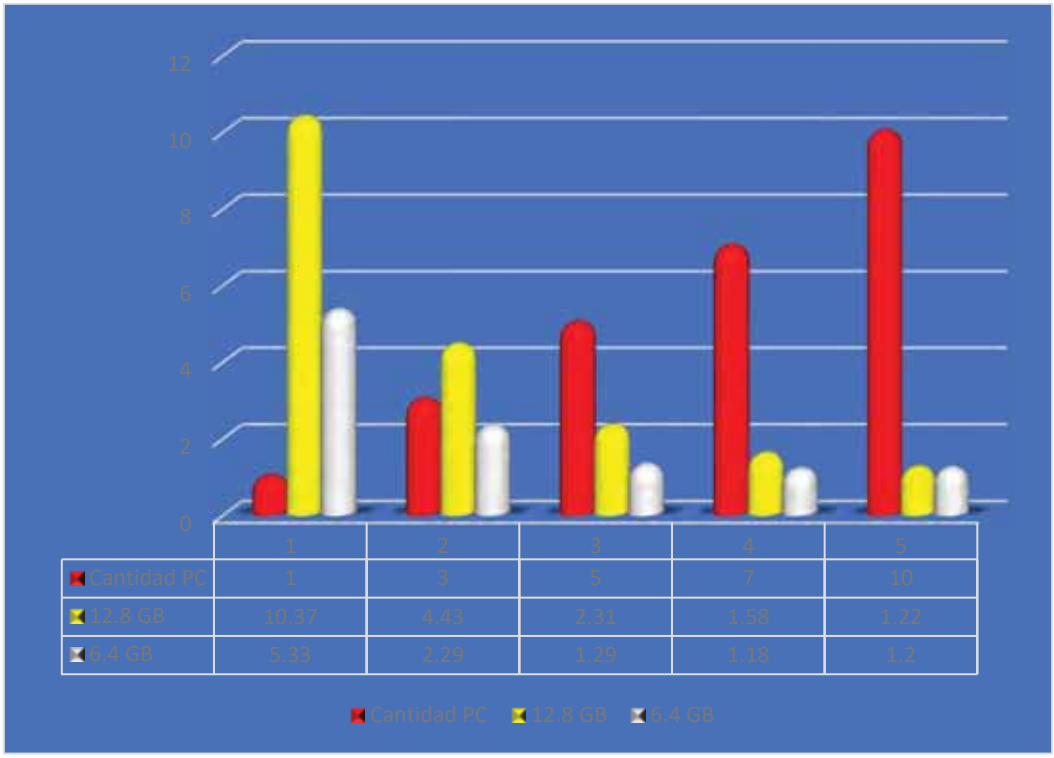


Figura 111. Rendimiento de 12.8 GB y 6.4 GB en clúster homogéneo (fuente propia)

5.3.8. Rendimiento en clúster heterogéneo

En primera instancia se realizó pruebas de rendimiento con 6.4 GB en clúster heterogéneo, pero para mejorar la estimación se realiza en segunda instancia con 12.8 GB.

Visualizar la representación en gráfico los tiempos de rendimiento generados por cantidad de nodos respectivas y, mostrar en barras de colores amarillos (12.8 GB) y blancos (6.4 GB) que descienden de más a menos.

La gráfica de barras rojas, indican la cantidad de nodos que ascienden de menos a más, referencia figura 113.

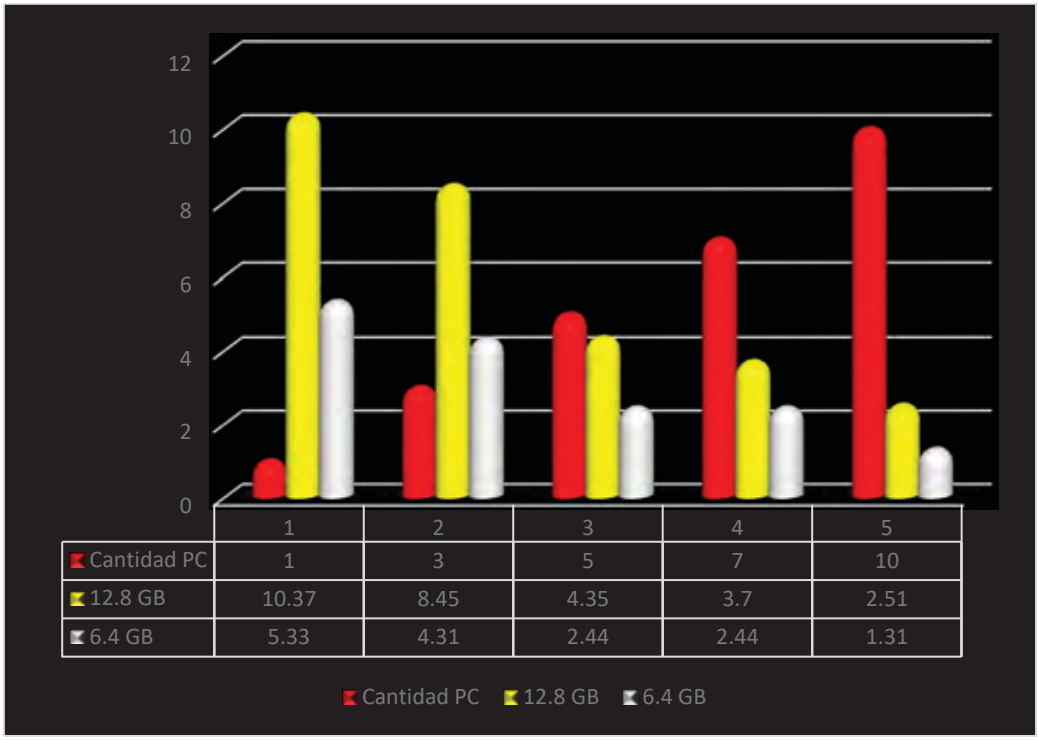


Figura 112. Rendimiento de 12.8 GB y 6.4 GB en clúster heterogéneo (fuente propia)

5.3.9. Contraste entre clúster homogéneo y heterogéneo.

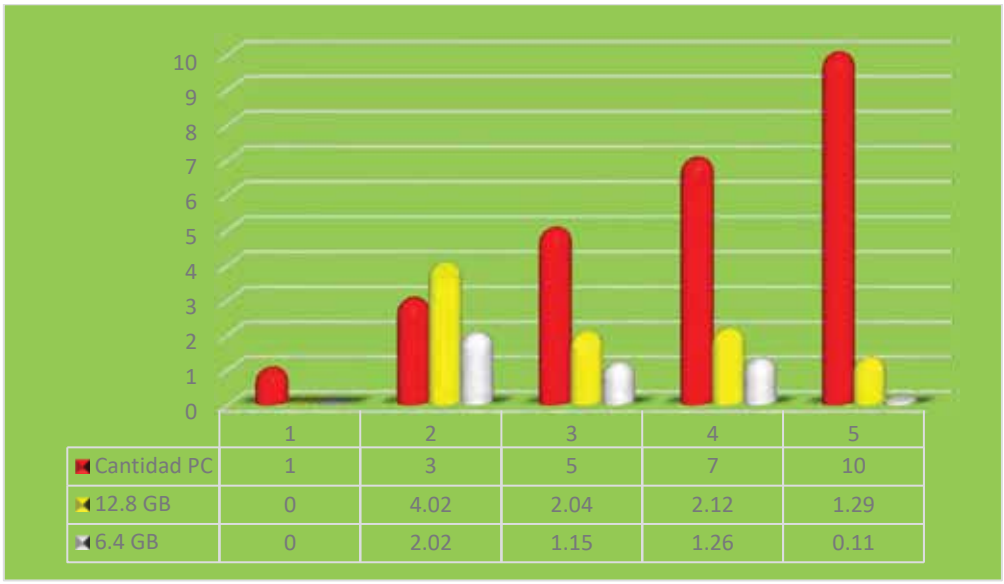
La diferencia de los tiempos de rendimiento entre clúster heterogéneo y homogéneo generados por cantidades de nodos, es la resolución siguiente.

El proceso de diferencia con 12.8 GB, entre clúster heterogéneo y clúster homogéneo, en detalle como sigue; Con 1 nodo la diferencia 0 segundos, con 3 nodos la diferencia de 4 min. y 2 seg, con 5 nodos la diferencia de 2 min y 4 seg, con 7 nodos la diferencia de 2 min y 12 seg y con 10 nodos la diferencia de 1 min y 29seg.

Se asevera la estimación, que el tiempo de rendimiento con 12.8 GB en clúster heterogéneo es casi el doble que en clúster homogéneo.

Así mismo con 6.4 GB la diferencia entre clúster heterogéneo y homogéneo, podemos visualizar la diferencia, que se mantiene el tiempo de rendimiento con mayor y menor grado de GB.

La estimación enmarcada general es que, mientras más nodos integran el clúster, el tiempo de rendimiento es muchos más óptimo, referencia figura 114.



	1	2	3	4	5
Cantidad PC	1	3	5	7	10
12.8 GB	0	4.02	2.04	2.12	1.29
6.4 GB	0	2.02	1.15	1.26	0.11

Figura 113. Diferencia entre clúster heterogéneo y homogéneo (fuente propia)

5.3.10. Discusión de resultados contra los antecedentes

- En este antecedente, Verma C, India (2016), en su trabajo de representación de grandes datos para el análisis de grado a través de Hadoop Framework, se hace énfasis en que la arquitectura MapReduce permite resolver problemas de grandes volúmenes de datos no estructurados en base a clave-valor. El presente trabajo coadyuva a esta idea y complementa en la posibilidad de efectuar este tipo de trabajos, estimando a priori tiempos y rendimientos de clúster en base a funciones matemáticas, con el consecuente beneficio de determinar las configuraciones más apropiadas para resolver el problema.

- En este antecedente, Sidhu, India (2016), en su trabajo de procesamiento por lotes eficientes de tareas relacionadas de Big Data utilizando la técnica de MapReduce persistente, se hace énfasis en el procesamiento por lotes eficiente y que la arquitectura MapReduce es apropiada para este propósito. El presente trabajo también complementa este propósito, porque también es factible ejecutar scripts simples, en arquitecturas homogéneas y heterogéneas; luego, también correlacionar estos resultados, para después proyectar rendimientos en base a la variación de nodos del clúster, que permitan configuraciones óptimas.
- En este antecedente, Ghazi, India (2015), en su trabajo de Hadoop, MapReduce y HDFS: Una perspectiva de desarrolladores, se hace énfasis en la necesidad de aplicar el framework de Apache Hadoop, en sus procesamientos de datos no estructurados y está distribuidos en HDFS, como dan uso las compañías de Facebook, Google, AWS, Microsoft y es apropiada para nuestro propósito, que proyectan los rendimientos y tiempos en base a cantidad de servidores commoditys heterogéneos y homogéneos y sus respectivas configuraciones.
- En este antecedente, Prabhu, India (2015), en su trabajo de mejora del rendimiento de Hadoop MapReduce FrameWork para analizar Big Data, se hace énfasis en la necesidad de aplicar las configuraciones del conjunto de parámetros de la arquitectura Apache MapReduce, una de ellas; Sobre las memorias, entrada y salida de disco, la red de los clúster homogéneas y heterogéneas y, proyectar los rendimientos y tiempos en base a mencionadas configuraciones, es apropiado para nuestro propósito.
- En este antecedente, Manikandan, India (2014), en su trabajo de análisis de datos grandes utilizando Apache Hadoop, se hace énfasis la relación con la configuración de las replicaciones de bloques por defecto 3, con la arquitectura

paralelo de MapReduce versión 1, que contienes a JobTracker y TaskTracker y, es muy apropiado para nuestro actual proyecto para MapReduce versión 2, conllevando proyectar los rendimientos y tiempos en base a cantidad de nodos en cada clúster homogéneo y heterogéneo.

- En este antecedente, Pal, India (2014), en su trabajo de un enfoque experimental hacia grandes datos para analizar la utilización de memoria en un clúster Hadoop utilizando HDFS y MapReduce, se hace énfasis en relación con nuestra arquitectura homogénea de para el tratamiento de datos no estructurados, sobre una instalación nativa. Conllevando a proyectar los rendimientos y tiempos en base a cantidad de nodos en los dos diferentes clústeres.
- En este antecedente, Mamta Padole, India (2018), en su trabajo de un equilibrio de carga a través de la política de reordenación de bloques para el clúster heterogéneo de Hadoop, se hace énfasis en relación con nuestra arquitectura heterogénea de para el tratamiento de datos no estructurados, sobre un clúster heterogéneo. Conllevando a proyectar los rendimientos y tiempos en base a cantidad de nodos en clúster heterogéneos.

CONCLUSIONES

1. Se logró diseñar la estructura de almacenamiento y procesamiento distribuido de datos no estructurados sobre Hadoop Distributed File System (HDFS). Dicho diseño ejecuta el proceso en paralelo con el modelo de programación MapReduce. El diseño ha consistido en un conjunto de 10 equipos con las mismas especificaciones para la estructura homogénea. Igualmente se ha considerado un conjunto de 10 equipos de diferentes especificaciones, para la estructura heterogénea.
2. La línea de tendencia, muestra la correlación de tiempos con respecto al número de nodos para las arquitecturas homogéneas y heterogéneas. Estos resultados muestran hacer proyecciones de grandes volúmenes de datos no estructurados, usando MapReduce y efectuando el almacenamiento en Hadoop Distributed File System (HDFS).

RECOMENDACIONES

1. Se sugiere llevar a la práctica lo planteado en el presente trabajo, para lo cual, aplicar a situaciones reales, donde se debe identificar organizaciones que procesen grandes volúmenes de datos no estructurados, y obtener a partir de ellos ventaja competitiva en la toma de decisiones.
2. Considerando el avance vertiginoso de la tecnología hardware, donde cada vez se cuenta con microprocesadores multinúcleo, efectuar estos mismos cálculos, en este tipo de hardware, analizando el comportamiento de los tiempos de procesamiento en equipos de diferente número de núcleos.
3. Considerando que a futuro se tendrá una dependencia inevitable de Big Data (Hadoop) con Ciencia de datos, Internet of Things, Maching Learning, Deep Learning e inteligencia artificial; se sugiere profundizar en Hadoop, que permita dar soluciones a problemas en estas áreas.
4. Lograr optimizar Hadoop aún más, con algoritmos de compresión, reduciendo más el espacio de HDFS y reducir el costo de transmisión de datos a través de la red.
5. Promover a las Empresas el uso de Hadoop que se ejecutan de manera shareware de Google.
6. Una característica de Hadoop, es escalar horizontalmente, tanto en clúster homogéneo y heterogéneo, en consecuencia, se optimiza el tiempo de procesamiento para tal unidad del dataset, sin embargo, para dar una certeza cuantitativa debemos aplicar la ley de Amdahl.

BIBLIOGRAFÍA

Aguilar, L. J. (2013). *Big Data: Análisis de grandes volúmenes de datos en organizaciones.* México: Alfaomega.

Alapati, S. R. (2017). *Administracion experta de Hadoop.* Estados Unidos de América: Pearson.

ALópez Borrull, A. C. (2013). *La Colaboración Científica en el marco de nuevas propuestas Científicas: BIG DATA.* Catalunya: Universidad Oberta de Catalunya.

American Psychological Association. (2010). *Manual de Publicaciones de la American Psychological Association* (6 ed.). (M. G. Frías, Trad.) México, México: El Manual Moderno.

Bhimani, J., & Leeser, M. (2017). *Aceleración de aplicaciones de big data utilizando un marco de virtualización ligero en la nube empresarial.* Waltham,USA: IEEE.

Dean, J., & Ghemawat, S. (2008). *MapReduce: procesamiento de datos simplificado en grandes grupos.* Nueva York: ACM.

Echevarría, R. R. (2014). *Estudio, análisis y evaluación del framework Hadoop.* Madrid: Universidad Pontificia Comillas.

Echevarría, R. R. (2014). *Estudio, Análisis y evaluación del Framework Hadoop.* Madrid.

Ghazi, M. R. (2015). *Hadoop, MapReduce and HDFS: A Developers Perspective.* India: ELSEVIER.

Jongyeop, K. (2015). *Identificación del conjunto óptimo de parámetros de configuración de Hadoop para computación Mapreduce.* Oklahoma State University: ieee.

Levin, R. R. (2004). *ESTADÍSTICA PARA ADMINISTRACIÓN Y ECONOMÍA.* México: Pearson Educación.

Locker, E. B. (2016). *Estudio de la aplicabilidad de algoritmos y tecnologías que presentan una red.* Venezuela: Universidad Central de Venezuela.

Madrid, U. P. (2015). *Big Data: el futuro a través de los datos.* Madrid: UPM.

Mai, P. D. (2016). *Tópicos Avanzados de Base de Datos.*

Manikandan, S. G. (2014). *Big Data Analysis using Apache Hadoop.* Tambaram, India: IEEE.

Masur, R., & McIntosh, S. (2017). *Análisis de rendimiento preliminar de Hadoop 3.0.0-alpha3.* New York.

Noelia, J. B. (2014). *Análisis y estudio de la plataforma Hadoop.* Madrid.

Pal, A. (2014). *An Experimental Approach Towards Big Data for Analyzing Memory Utilization on a Hadoop cluster using HDFS and MapReduce.* Guntur, India: IEEE.

Prabhu, S. (2015). *Performance Enhancement of Hadoop MapReduce Framework for Analyzing BigData .* India: IEEE.

Purshotam, S. (2013). *Adaptive Application Master para Elastic Web Server.* China, Fuzhou: IEEE.

Rattanaopas, K., & Kaewkeeree, S. (2017). *Mejora del rendimiento de Hadoop MapReduce con compresión de datos: un estudio con Wordcount Job.* Thailandia: IEEE.

Rodríguez, S. L. (2014). *Big Data, Hadoop y DataMinig sobre eventos de seguridad.* Sevilla.

Sachdeva, K., & Kaur, J. (2017). *Procesamiento de imágenes en clúster Hadoop de nodos múltples.* India: IEEE.

Salazar, C. (2018). *Fundamentos básicos de estadística.*

Schaum, M. S. (1977). *Probabilidad y estadística .* Bogotá-Colombia.

Shah, A., & Padole, M. (2018). *Equilibrio de carga a través de la política de reorganización de bloques para el clúster heterogéneo de Hadoop.* India: IEEE.

Sidhu, R. K. (2016). *Efficient Batch Processing of Related Big Tasks using Persistent MapReduce.* India: ACM.

Verma, C. (2016). *Big Data representation for Grade Analysis Through.* 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence).

Verma, C. (2016). *Big Data representation for Grade Analysis Through.*

Vishal, A., & Sai Raghu Ram, G. (2017). *Comparación de rendimiento de Hadoop y Spark Engine.* India: IEEE.

Sanjay Agrawal (2014). AN EXPERIMENTAL APPROACH TOWARDS BIG DATA FOR ANALYZING MEMORY UTILIZATION ON A HADOOP CLUSTER USING HDFS AND MAPREDUCE. First International Conference on Networks & Soft Computing

A López Borrull, A Canals (2013). LA COLABORACIÓN CIENTÍFICA EN EL MARCO DE NUEVAS PROPUESTAS CIENTÍFICAS: BIG DATA. Universitat Oberta de Catalunya.

Melanie Swan (2015). PHILOSOPHY OF BIG DATA. First International Conference on Big Data Computing Service and Applications

Jean-Pierre (2013). ORACLE: BIG DATA FOR THE ENTERPRISE RED WOOD: ORACLE ENTERPRISE

Revista 2015 Universidad Politécnica de Madrid.

Augsburger Becerra, M. A. (2014) PARALELIZACIÓN DE UN ALGORITMO PARA LA DETECCIÓN DE CÚMULOS Universidad de Chile Facultad de Ciencias Físicas y Matemáticas Departamento de Ciencias de la Computación.

T. White, "Hadoop. The Definitive Guide" O Reilly, 2da edición, pp. 27-30, Octubre 2012.

Rubio Echevarría, R. (2014) ESTUDIO, ANÁLISIS Y EVALUACIÓN DEL FRAMEWORK HADOOP Universidad Pontificia Comillas-Madrid.

Domínguez Romero, E. (2015) SOFTWARE ENGINEER RIPE NCC Ámsterdam y alrededores, Países Bajos.

Anexo1

UnsaacMapReduce.java

```java
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
/*modelo UnsaacMapReducer*/
public class UnsaacMapReducer {
/*mapper*/
 public static class TokenizerMapper
 extends Mapper<Object, Text, Text, IntWritable>{

 private final static IntWritable one = new IntWritable(1);
 private Text word = new Text();

 public void map(Object key, Text value, Context context
 ) throws IOException, InterruptedException {
 StringTokenizer itr = new StringTokenizer(value.toString());
 while (itr.hasMoreTokens()) {
 word.set(itr.nextToken());
 context.write(word, one);
                        }
                                    }
                                        }
 /*reducer */
 public static class IntSumReducer
 extends Reducer<Text,IntWritable,Text,IntWritable> {
 private IntWritable result = new IntWritable();
 public void reduce(Text key, Iterable<IntWritable> values,
 Context context
 ) throws IOException, InterruptedException {
 int sum = 0;
 for (IntWritable val : values) {
 sum += val.get();
                        }
 result.set(sum);
 context.write(key, result);
                                    }
                                            }
 /*La entrada de cualquier fichero*/
 public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
 if (otherArgs.length != 2) {
 System.err.println("Uso: ContarPalabras <in> <out>");
 System.exit(2);
                        }
 Job job = Job.getInstance(conf, "ContarPalabras");
 /*Generar clases*/
 job.setJarByClass(UnsaacMapReducer.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
 FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
 System.exit(job.waitForCompletion(true) ? 0 : 1);
                                            }
}
```