FACULTAD DE INGENIERIA Y CIENCIAS APLICADAS

ANÁLISIS DE PLATAFORMAS DE BIG DATA APLICADAS A UN CAMPUS INTELIGENTE

AUTOR

Jhoann Sebastián Molina Enríquez

AÑO

2019

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

ANÁLISIS DE PLATAFORMAS DE BIG DATA APLICADAS A UN CAMPUS INTELIGENTE.

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor Guía

MSc. William Eduardo Villegas Chiliquinga

Autor

Jhoann Sebastián Molina Enríquez

Año

2019

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Análisis de plataformas Big Data aplicadas a un campus inteligente, a través de reuniones periódicas con el estudiante Jhoann Sebastián Molina Enríquez, en el semestre 201920, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

William Eduardo Villegas Chiliquinga

Magister en Redes de Comunicaciones

C.I.: 1715338263

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Análisis de plataformas Big Data aplicadas a un campus inteligente, del Jhoann Sebastián Molina Enríquez, en el semestre 201920, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Iván Patricio Ortiz Garcés

Magister en Redes de Comunicaciones

C.I.: 0602356776

DECLARACIÓN DE AUTORIA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

_____

Jhoann Sebastián Molina Enríquez

C.I.: 1719089383

AGRADECIMIENTOS

Agradezco a Dios por bendecirme a lo largo de mi carrera, por todas las cosas maravillosas que me da cada día. A mis padres por su dedicación, esfuerzo y amor para verme realizado como profesional. A mi hermano, por preocuparse por su hermano mayor y estar en cada momento importante de mi vida. A mi novia, por su amor, paciencia, su incondicional apoyo en las situaciones más difíciles que forma parte esencial de la culminación de mi carrera. Un sincero agradecimiento a mi tutor de tesis, por su invaluable apoyo, consejos, experiencia y respaldo que ha sido de mucha importancia en la realización de esta tesis. Gracias a todas las personas que formaron parte de mi vida y de mi carrera, este logro, también es de ustedes, gracias por creer en mí.

DEDICATORIA

A mis padres y mi hermano que son mi bendición, que con su esfuerzo siempre me apoyaron con su incondicional amor. A mi novia que siempre estuvo apoyándome y motivándome para poder cumplir mis metas. A todos mis compañeros y profesores que formaron parte de mi crecimiento como profesional.

RESUMEN

El presente trabajo de titulación tiene como objetivo realizar un análisis detallado de las plataformas de Big Data que se podrían implementar en un campus inteligente.

El desarrollo de este trabajo se enfoca a un análisis de las diferentes plataformas de Big Data y que sirva como guía o referencia para que el personal de un campus universitario tenga una visión mucho más clara sobre que plataforma sería la mejor para implementarla en un campus universitario.

En este proceso de análisis se desarrollará una guía para la implementación de plataformas de Big Data. Este trabajo constará de varias partes. En la primera, introducción, se presentará las generalidades.

El segundo tema demuestra el cuerpo de trabajo que está dividido en varios temas como definiciones de Big Data y de sus diferentes plataformas, el análisis de los datos, luego se seleccionará y se analizará cada plataforma escogida las cuales serán comparadas y de esa manera poder justificar sus mejores características para su respectiva implementación en un campus inteligente en un futuro. Este documento busca que los lectores conozcan las ventajas de la implementación de una plataforma para lograr una arquitectura escalable.

Finalmente, en la tercera parte se presenta es la conclusión del documento que sirva como una guía detallada y que pueda servir de respaldo cuando se requiera escoger una plataforma para un ambiente Big Data que cumpla con las necesidades de un campus inteligente.

ABSTRACT

The objective of this thesis work is to carry out a detailed analysis of the Big Data platforms that could be implemented in an intelligent campus.

The development of this thesis work focuses in the analysis of different Big Data platforms to be useful and serves as a guide or reference for the personnel of a university with the objective that they could have a clear vision about which platform would be the best to implement it on a university campus.

In this analysis process, a guide will be developed for the implementation of the Big Data platforms. This work will consist of several parts. In the first part, the introduction, the generalities will be presented.

The second topic presents the body of work that is divided in several topics as Big Data definitions and of its different platforms, analysis of data. Then, it will be select and analyze each chosen platform. In this way, the platforms will be compared to justify their best characteristics for the respective implementation in this intelligent campus in the future. This document seeks that readers know the advantages of the platform implementation to achieve a scalable architecture.

Finally, in the third part there will present the conclusion of the document. That serves as a detailed guide that could be useful to support when it requires to choose a platform for a Big Data ambiance that accomplish with the requirements of an intelligent campus.

ÍNDICE

1. Introducción 1

1.1 Alcance 2

1.2 Justificación 2

1.3 Objetivo General 3

1.4 Objetivos Específicos 3

1.5 Metodología 4

2. Marco Teórico 4

2.1 Big Data 4

2.1.1 Smart Data 5

2.1.2 Data Lake 5

2.1.3 Evolución de Big Data 6

2.1.4 Las 7 V del Big Data 7

2.1.4.1 Volumen de la información 7

2.1.4.2 Velocidad de los datos 8

2.1.4.3 Variedad de los datos 8

2.1.4.4 Veracidad de los datos 9

2.1.4.5 Viabilidad 9

2.1.4.6 Visualización de los datos 9

2.1.4.7 Valor de los datos 9

2.2 Arquitectura de Big Data 11

2.2.1 Ingreso de datos 13

2.2.2 Gestión de datos 13

2.2.3 Análisis de datos 13

2.2.4 Tiempo real de procesamiento 13
2.3 Tipos de datos 13
2.3.1 Datos Estructurados 13
2.3.2 Datos no Estructurados 14
2.3.3 Datos Semiestructurados 15
2.4 Bases de Datos 16
2.4.1 Gestores de Bases de Datos 16
2.4.2 Bases de Datos Relacionales 16
2.4.3 Sistemas de gestores de Bases de Datos 16
2.4.6 Modelos de datos no relacionales 17
2.4.6.1 Estructura del esquema de información 17
2.4.6.2 Guardado en clave y valor 18
2.5 Minería de datos 19
2.5.1 Proceso de KDD 19
2.5.2 Técnicas de minería de datos 21
2.5.2.1 Técnicas predictivas 21
2.5.2.2 Técnicas descriptivas 21
2.6 Internet of Things (IoT) 21
2.7 Smart Campus 22
2.7.1 Aplicación del Internet de las cosas y Cloud Computing 23
2.7.1.1 Internet de las cosas en la educación 23
2.7.1.2 Cloud computing e Internet de las cosas 24
2.7.2 Características de un Campus Inteligente 26
2.7.3 Componentes de un Smart Campus 27
3. Tecnologías y Plataformas de Big Data 28

3.1 Apache Hadoop 28
3.1.1 Características básicas 29
3.1.1.1 Cuando usar Hadoop 30
3.1.1.2 Cuando no usar Hadoop 31
3.1.2 Arquitectura básica, módulos de Hadoop 32
3.1.3 Proceso de Hadoop 32
3.1.4 MapReduce 33
3.1.5 Analizar datos con Hadoop 34
3.1.5.1 Java MapReduce 34
3.1.5.2 Prueba de funcionamiento 34
3.1.4.2 Escalamiento 35
3.1.5 HDFS 36
3.1.6 Componentes de HDFS 37
3.1.6.1 Bloques 37
3.1.6.2 Flujo de datos 37
3.1.6.3 Proceso de un archivo leído 38
3.1.6.4 Proceso de un archivo escrito 38
3.1.7 YARN 41
3.1.7.1 Arquitectura de YARN 42
3.1.8 Componentes de YARN 43
3.1.8.1 Administrador de recursos 43
3.1.8.2 Aplicación Master 44
3.1.8.3 Administrador de Nodos 44
3.1.8.4 Contenedor 45
3.1.8.5 Ambiente de desarrollo 46
3.2 Apache Spark 46

3.2.1 Características principales 47
3.2.2 Librerías Externas 47
3.2.4.1 Serialización de RDD 48
3.2.5 Componentes de Spark 50
3.2.5.1 Spark Core 51
3.2.5.2 Spark SQL 51
3.2.5.3 Spark Streaming 51
3.2.5.4 Spark MLLib 51
3.2.5.5 Spark Graph 51
3.2.6 Velocidad de procesamiento 52
3.2.7 Funcionamiento en cualquier parte 52
3.3 Arquitectura Spark 53
3.3.1 API de Spark 54
4. Análisis de los resultados 54
4.1 Análisis de las plataformas Hadoop y Spark 55
4.1.1 Arquitectura 56
4.1.2 Posición en la industria 56
4.1.2 Velocidad del proceso 58
4.1.2 Rendimiento 59
4.1.3 Usabilidad 60
4.1.4 Costos 62
4.1.6 Procesamiento de datos 64
4.1.7 Tolerancia a fallos 66
4.1.8 Seguridad 67
4.1.9 Escalabilidad 68

4.2 Diferencias entre Hadoop y Spark 69
5. Conclusiones y Recomendaciones 70
5.1 Conclusiones 70
5.2 Recomendaciones 71
Referencias 73

ÍNDICE DE TABLAS

Tabla 1. Análisis de la capacidad de un conjunto de datos. 6

Tabla 2. Datos enviados por aplicaciones en el año 2016..... 8

Tabla 3. Atributos de Big Data. 10

Tabla 4. Ejemplo de datos estructurados (Structured Data). 14

Tabla 5. Ejemplo almacenamiento como parejas de clave y valor..... 18

Tabla 6. Diferencias entre un campus digital y un campus inteligente..... 25

Tabla 7. Principales características de Hadoop..... 28

Tabla 8. Mezcla de modelos distribuidos para la creación de Hadoop. 29

Tabla 9. Sistema de archivos Hadoop..... 41

Tabla 10. Principales características de Spark..... 46

Tabla 11. Características MapReduce y Spark. 55

Tabla 12. Comparación de la velocidad de procesamiento. 59

Tabla 13. Formato del control de costos. 64

Tabla 14. Comparación entre Hadoop y Spark..... 68

ÍNDICE DE FIGURAS

Figura 1. Las “V” de Big Data. 11

Figura 2. Arquitectura de un ambiente de Big Data. 12

Figura 3. Ejemplo de datos no estructurados (Unstructured Data). 15

Figura 4.Ejemplo de datos semiestructurados (Semistructured Data). 15

Figura 5. SQL (Structured Query Language). 17

Figura 6. Ejemplo de esquema anidado de información. 18

Figura 7. Minería de datos. 19

Figura 8. Minería de datos – proceso KDD. 20

Figura 9. Sistema bajo el entorno IoT. 23

Figura 10. Plataforma del sistema de educación en la nube. 24

Figura 11. Rendimiento de la red con dispositivos conectados. 25

Figura 12. Campus Internet de las cosas IoT. 26

Figura 13. Componentes considerados para un Smart Campus. 27

Figura 14. Logotipo Hadoop. 28

Figura 15. Cuando usar Hadoop. 30

Figura 16. Cuando no usar Hadoop. 31

Figura 17. Arquitectura de Hadoop. 32

Figura 18. Esquema del proceso de Hadoop. 33

Figura 19. Funcionamiento de MapReduce. 34

Figura 20. Proceso de ejecución del trabajo. 35

Figura 21. Diagrama de procesos de flujo de datos. 39

Figura 22. Modelo clásico y actual. 42

Figura 23. Arquitectura de YARN. 43

Figura 24. Administrador de recursos. 44
Figura 25. Aplicación Master. 44
Figura 26. Administrador de nodos. 45
Figura 27. Beneficios de YARN. 45
Figura 28. Logotipo de Apache Spark. 46
Figura 29. Tipos de transformaciones Narrow y Wide. 50
Figura 30. Estructura componentes principales de Spark. 50
Figura 31. Regresión logística en Hadoop y Spark 52
Figura 32. Arquitectura de Apache Spark. 53
Figura 33. Desarrolladores de Spark y Hadoop. 57
Figura 34. Histórico de aportaciones en OpenHub. 57
Figura 35. Histórico de aportaciones en GitHbub. 57
Figura 36. Lenguajes de programación que usan Spark. 58
Figura 37. Impacto e interés Spark vs Hadoop. 61
Figura 38. Componentes de un modelo de gestión de costos. 62
Figura 39. Capacidad de servicios tradicionales con servicios modernos. 63
Figura 40. Proceso Spark Streaming. 65

1. Introducción

Actualmente las tecnologías de la información permiten acceder y generar grandes volúmenes de información, con el objetivo de simplificar procesos y convertir datos en información. Los resultados esperados al implementar los servicios de una plataforma dependen de un análisis correcto de necesidad y beneficio y elaborando un documento científico; para poder garantizar buenos resultados en corto, mediano y largo plazo.

En los últimos años el termino inteligente está posicionado a nivel mundial, los dispositivos y sistemas inteligentes como dispositivos electrónicos, hogares, edificios, campus pueden ajustarse según los cambios tecnológicos. Las plataformas basadas en sistemas inteligentes no solo pueden realizar ajustes sino también pueden aprender sobre los usuarios y su entorno (Aion, Helmandollar, Wang, & Ng, 2012).

Cuando se habla de Big Data se refiere a millones de datos que se generan por segundo. El manejo de cantidades tan grandes de datos conlleva a pensar en implementar plataformas de Big Data que permitan generar, almacenar y procesar datos. Después se tiene como finalidad usar estos datos transformados en información para la búsqueda de patrones y tomar decisiones oportunas. En la actualidad existen varias plataformas que se podrían implementar en un campus universitario.

La educación utiliza plataformas para el aprendizaje de los estudiantes, mediante esta integración se puede gestionar el aprendizaje y crear nuevos métodos. Estas plataformas se han convertido en una fuente muy importante de datos se busca integrar los recursos académicos que los estudiantes utilizan para el desarrollo de sus actividades.

Las universidades están sujetas a controles por parte de entidades gubernamentales y organismos que certifiquen la garantía de una buena educación, pero para las universidades resulta complicado saber exactamente las causas reales de la deserción estudiantil. Actualmente según estudios

realizados se puede saber las causas y detectar variables que indiquen las razones de deserción.

Considerando la necesidad de las instituciones de educación superior de conocer el motivo por el cual existe mayor deserción de los estudiantes y brindar una mejor educación se ha llevado a cabo este documento el cual está estructurado en capítulos teniendo como objetivo determinar la o las mejores plataformas de Big Data que pueden ser aplicadas a un campus inteligente.

1.1 Alcance

En este documento se analiza las oportunidades que ofrece Big Data para mejorar el aprendizaje con un sistema de monitoreo a los estudiantes. Mediante este análisis se desea incentivar a las universidades que busquen convertirse en grandes bancos de datos, cual herramienta va a brindar las mejores especificaciones. Este análisis servirá para que el equipo que desee implementar Big Data en el campus de una universidad y tendrá un documento en el cual se podrá guiar y encontrar la información necesaria para saber cuál es el mejor para sus necesidades.

En el proceso de análisis se busca brindar información útil y contribuir al mejoramiento de la educación. Para que las autoridades universitarias tomen decisiones y existe una menor tasa de deserción por parte de los estudiantes.

1.2 Justificación

Teniendo en cuenta la necesidad de tratar el fenómeno de la deserción estudiantil se realizó este análisis de las plataformas de Big Data que pueden ser implementadas en un campus universitario.

El tema de las plataformas de Big Data aplicables a un campus inteligente es importante para conocer como Big Data transforma los procesos educativos.,

con la investigación se busca cambiar los modelos tradicionales por nuevos servicios que están emergiendo en la actualidad.

Un beneficio de la investigación es contar con un documento que abarque la investigación de plataformas de Big Data que aporten a la implementación de modelos de análisis de datos y que faciliten las técnicas que permitan descubrir las tendencias de los alumnos, docentes y personal administrativo. Los resultados de la investigación contribuirán a la creación de métodos de enseñanza nuevos y potenciar al máximo las capacidades de las personas que lo usan.

Debido a los índices de deserción estudiantil se requiere tener un estudio previo que pueda ser aplicable a un campus universitario con la finalidad de tener un modelo integrado mediante una plataforma de Big Data capaz de analizar y procesar los datos generados y tomar buenas decisiones.

1.3 Objetivo General

Realizar un análisis comparativo de las plataformas de Big Data aplicado a un campus inteligente.

1.4 Objetivos Específicos

- Identificar las tecnologias y plataformas de software con sus requeriminetos necesarios para la implemetación de una plataforma de Big Data en un campus inteligente.
- Identificar el ordenamiento formal carcterístico de cada plataforma seleccionada para el análisis.
- Determinar la infliuencia especifica de las plataformas para su implementación en un campus inteligente.

1.5 Metodología

Para el tratamiento del material de la investigación se empleará el método inductivo, porque es necesario sustentar con diversos análisis de las diferentes plataformas que existen de Big Data.
Se realizará la recopilación de la información de cada plataforma y un análisis con algunas herramientas de comparación. Los recursos usados básicamente serán las páginas de los fabricantes, documentos y se tendrá que tener familiarización con los softwares seleccionados. Mediante simulaciones elaborar el documento con información real y confiable. Mediante la información adquirida se busca llegar a tener ideas concretas y de esa manera tomar una resolución de cuál es la plataforma más conveniente.

2. Marco Teórico

2.1 Big Data

Son volúmenes importantes de información cuyas necesidades de almacenamiento y procesamiento no pueden ser atendidas por las tecnologías tradicionales que normalmente se usan en sistemas de información como bases de datos relacionales.
El termino Big Data significa literalmente datos masivos. Se define como la gestión y el análisis de enormes volúmenes de datos que no pueden ser tratados de una manera que comúnmente se trasladan. Estos datos superan la capacidad y los límites de las herramientas de software usadas para el tratamiento, procesamiento y análisis de datos. El principal objetivo es convertir estos datos en información para poder tomar decisiones en corto plazo, largo plazo o en tiempo real (Demchenko, 2013).

Las empresas o universidades suelen utilizar estos entornos para saber y entender el perfil de los usuarios o estudiantes en el caso de las universidades

para saber cómo se siente con respecto a los productos o servicios que ofrecen. (MicroStrategy, 2019).

2.1.1 Smart Data

El concepto de Smart Data tiene su aparición a través de Big Data. Centrándose en el procesamiento y la transformación de una gran cantidad de datos en información útil. En otras palabras, se puede decir que Big Data se encarga de almacenar y procesar la información, una vez obtenida dicha información el Smart Data mediante el uso de fórmulas matemáticas realiza un proceso para convertir estos datos en respuestas (Rocha y Guarda, 2018).

2.1.2 Data Lake

El término lago de datos se refiere a un gran repositorio de almacenamiento en el que los datos en bruto se mantienen almacenados en su formato original hasta que estos sean necesarios. Contiene un gran conjunto de datos que no tienen definida una estructura. El principal beneficio de un lago de datos es la centralización de fuentes de contenido dispares, una vez ensambladas esas fuentes se pueden combinar y procesar utilizando Big Data (Buenaño-Fernandez, Villegas-CH y Luján-Mora, 2019).

Características de Big Data, Smart Data y Data Lake

En la tabla 1 se muestra una comparación de las características técnicas de Big Data, Smart Data y Data Lake, tomando en cuenta que se analizó cada plataforma según su capacidad de trabajo, por ejemplo, Big Data se centrara en la capacidad que tiene para gestionar un volumen de datos muy grande, Smart Data según su función que es actuar como un filtro de datos y Data Lake en compartir sus características de un almacén de datos.

Tabla 1.

Análisis de la capacidad de un conjunto de datos

Plataforma	Capacidad	Fuente	Costo de almacenamiento
Big Data	Conjunto de datos muy grande	Conjunto de datos complejos	Alto
Smart Data	Filtro de datos	Datos en bruto	Medio
Data Lake	Mejor versión del almacén de datos	Datos en bruto	Bajo

2.1.3 Evolución de Big Data

A continuación, se presenta cronológicamente estadísticas referente al crecimiento de los datos. En este caso el exceso de información se produjo en 1880, tras la problemática desarrollada en ese tiempo, derivó a la creación de la tabuladora de Hollerith. Tiempo después llego el “Boom del crecimiento demográfico” en 1932, esto quiere decir que en Estados Unidos toda la información que generaba su población llevó a una sobrecarga de la misma. (Winshuttel, 2019).

La primera vez que se presentó el problema del almacenamiento y la recuperación de datos fue en 1944 cuando Fremont Rider, bibliotecario de la Universidad de Wesleyana. Calculó que las bibliotecas de todas las universidades de Estados Unidos duplicaban su tamaño cada 16 años. Surgió “La teoría de información de Shannon”, (Monsalve, 2003, p. 23) la actualidad es la base del procesamiento de datos moderno. En 1983 las empresas comenzaron a utilizar el procesamiento de datos para tomar decisiones.

En 1989 aparece la inteligencia empresarial o business intelligence, este método utilizaba las empresas para analizar sus datos. (Winshuttle, 2019).

El problema de Big Data apareció en julio de 1997 a medida del crecimiento de los datos se tornaba un problema para los sistemas informáticos. Se decidió cuantificar la información teniendo como resultado que el mundo produjo 1,5 exabytes. Apareció la era del Internet de las cosas, *Internet of Think (IoT)*. En febrero del 2001 aparecen las V de Big Data.

En el 2006 aparece Hadoop como una solución de código abierto para la explosión del Big Data, esta plataforma sirve para almacenar y procesar os datos. Se produjo una nueva explosión de datos que podría alcanzar el zettabyte. Aparición del ERP en la nube. Ciudades inteligentes. El futuro del Big Data se calcula un aumento estimado de 4300% en la generación de datos para el 2020. (Winshuttle, 2019)

2.1.4 Las 7 V del Big Data

Normalmente Big Data se enfoca hacia la resolución de problemas de datos y busca hacer eficiente a los atributos que forman las principales características como:

2.1.4.1 Volumen de la información

Es la cantidad de datos que se genera cada segundo y se almacenan y procesar esos datos para que se transformen en información y posteriormente en acciones. Se calcula que la cantidad de información que se generó en el mundo entre el comienzo de los tiempos y el año 2008 será comparable en tamaño a la que se generará cada minuto a partir de ahora (Camargo-Vega, Camargo-Ortega y Joyanes-Aguilar, 2015). En el 2016 se calcula que cada minuto en Internet se enviaron:

Tabla 2.

Datos enviados por aplicaciones en el año 2016

Cantidad	Aplicación
204000000	Mensajes de correo electrónico
4500000	Me gusta en Facebook
350000	Tweets en Twitter
2500000	Búsquedas en Google
2000000	Likes en Instagram

Tomado de: Camargo-Vega, Camargo-Ortega y Joyanes-Aguilar, 2014

2.1.4.2 Velocidad de los datos

Se refiere a la rapidez en la que los datos son creados, almacenados y procesados los datos en movimiento es decir en tiempo real y velocidad a la que se trasmite e intercambia esta información. Un ejemplo real es como hoy en día se puede distribuir una foto que se toma a millones de personas por todo el mundo a través de las redes sociales o lo rápido que Google puede devolver información del otro lado del planeta al realizar una búsqueda. (Camargo-Vega et al. 2015).

2.1.4.3 Variedad de los datos

Se refiere a las formas, tipos y fuentes en las que los datos son registrados. Estos pueden se estructurados que resulta fáciles de gestionar como las bases

de datos y los no estructurados como documentos de texto, correos electrónicos, audio, video para estos se requiere una herramienta especifica.
Hasta hace poco la información que había en los sistemas tecnológicos era estructurada, es decir, seguía patrones específicos para ser almacenada en sistemas digitales. En la actualidad más del 80% de los datos que hay en el mundo son no estructurados. (Camargo-Vega et al. 2015).

2.1.4.4 Veracidad de los datos

Se refiere a la incertidumbre que se produce al almacenar y procesar los datos que recibe, estos datos recolectados deben ser confiables para realizar el debido procesamiento de los mismos. Es la autenticidad, calidad, fiabilidad de información, no toda la información que se genera es precisa o verdadera y uno de los retos de Big Data es poder identificar que datos son ciertos cuales no lo son y desecharlos. (Esumer, 2017).

2.1.4.5 Viabilidad

Es la capacidad para visualizar que la implementación de Big Data sea factible y tenga éxito. Es decir que busca mirar que tan viable es implementar una tecnología.

2.1.4.6 Visualización de los datos

Es la manera de cómo se presentan los datos de manera entendible para que se pueda interpretar la información.

2.1.4.7 Valor de los datos

El valor obtenido son los datos recolectados y procesados que se transforman en información valiosa y útil para poder tomar decisiones. La información es importante porque tiene y genera un valor el análisis de los datos se debe

realizar con estrategias de alto rendimiento. Actualmente la mayor fuente de información son las personas, nosotros mismos de toda la información que se genera a diario el 75% es generada por las personas.

Tabla 3.
Atributos de Big Data

Volumen	Variedad	Velocidad	Veracidad	Valor
Almacenamiento en terabytes	Por lotes	Estructurado	Integridad y Autenticidad	Estadísticas
Registros	Tiempo cercano	No Estructurado	Origen y Reputación	Eventos
Transacciones	Tiempo Real	Multi-factor	Disponibilidad	Correlación
Tablas y Archivos	Procesos	Probabilística	Responsabilidad	Hipótesis

A diferencia de los sistemas convencionales que usaban fuentes de datos estructuradas como las bases de datos, el Big Data obtiene los datos de diversas fuentes incluyendo datos.

Estas siete V en las que se basa Big Data que son los atributos y propiedades de los datos también se puede explicar como se muestra en la figura 1.

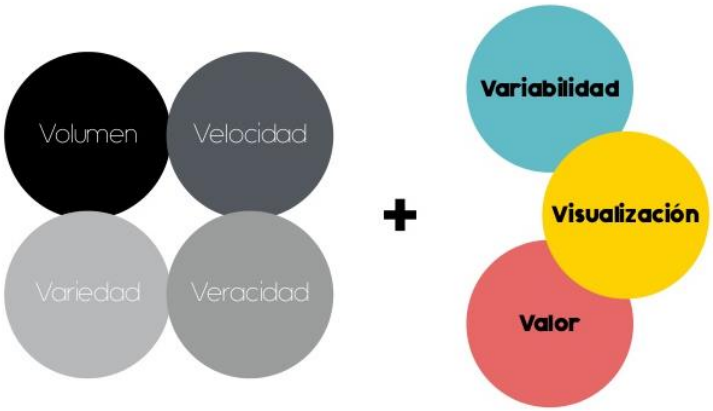


Figura 1. Las “V” de Big Data.
Tomado de (Ingeniare, 2015)

Para comprender mejor se considera otro punto considerado en Big Data son los procesos que se mencionan a continuación:

- Recogida de información de diferentes fuentes
- Integración de la información
- Almacenamiento de datos
- Validación de datos
- Análisis
- Explotación

2.2 Arquitectura de Big Data

Un ambiente de Big Data maneja con el diseño de una arquitectura de 5 niveles. En el siguiente gráfico se puede visualizar los niveles de Big Data y de qué manera interactúan.
Para hacer una arquitectura de datos se debe combinar componentes que varían y que sus características cambian según la necesidad de cada proyecto para que cada componente se relacione con todas las fases de la arquitectura.

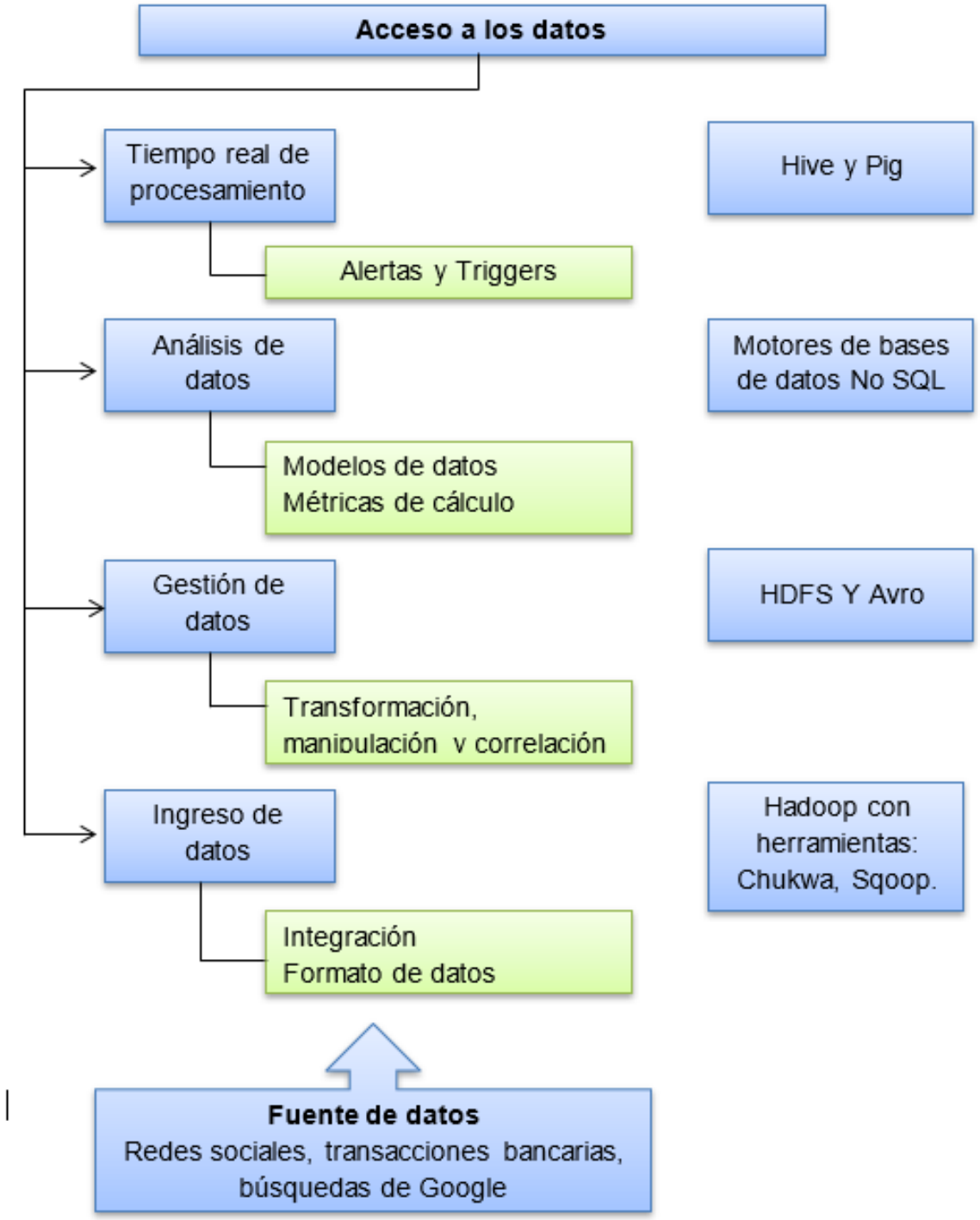


Figura 2. Arquitectura de un ambiente de Big Data.
Adaptado de (University of Amsterdam, 2013)

2.2.1 Ingreso de datos

En este punto se obtiene la información receptada de diferentes fuentes como transacciones, búsquedas, etc., para analizarla y que sea guardada en una base de datos.

2.2.2 Gestión de datos

Los datos son gestionados con la implementación de políticas y otros métodos para que este proceso por el cual pasan los datos para convertirse en información útil. Se pretende administrar los datos sean estructurados o no estructurados a gran escala.

2.2.3 Análisis de datos

Consiste en analizar los datos ya procesados y transformados en información, en este paso se busca si existen relaciones, modelos de datos, es decir información que será de utilidad.

2.2.4 Tiempo real de procesamiento

Según la figura 2 es un proceso que trata de automatizar los datos obtenidos para obtener infromación, aprovechando el flujo de los datos para penetrar a la información estática.

2.3 Tipos de datos

La importancia de los tipos de datos que se van a analizar son importantes en la realización de cualquier proyecto, especialmente en Big Data ya que se trabaja con un gran volumen de datos.

2.3.1 Datos Estructurados

La información está estructurada, quiere decir que tiene un esquema, básicamente es cómo funcionan las bases de datos relacionales estilo bases

de datos SQL. Donde sí se tiene una tabla de usuarios o de facturas en estos casos se conoce la estructura de la información es decir que posee campos fijos. Tienen bien definido la longitud, el formato y el tamaño de sus datos. (Fabian Guerrero, 2013).

Los datos pueden ser:

- **Creados:** se generan de una manera definida, por ejemplo, los ficheros XML.
- **Provocados:** son datos que se crean de manera indirecta por una acción generada anteriormente.
- **Por transacciones:** son datos que se generan después de realizar alguna acción anteriormente por ejemplo el recibo de un cajero.
- **Compilados:** son los resúmenes de los datos obtenidos después de un estudio como un censo.
- **Experimentales:** son datos de pruebas para verificar la viabilidad.

Tabla 4.

Ejemplo de datos estructurados (Structured Data)

	Nombre	Color	Edad	Altura	Peso	Puntuación
1	Paco	Rojo	24	182	74.2	83
2	Juan	Amarillo	62	169	60.8	269
3	Andrés	Verde	32	175	88.9	65
4	Juan	Negro	21	170	66.5	811

2.3.2 Datos no Estructurados

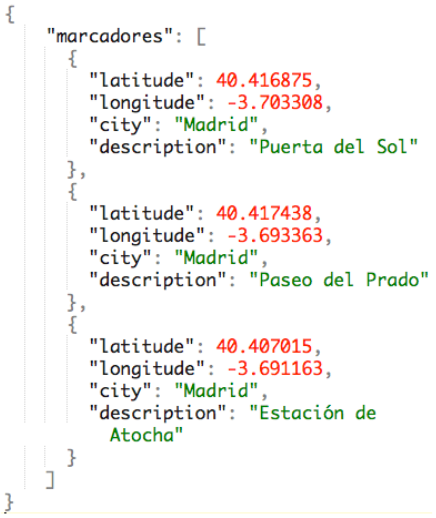
Los datos que no están estructurados pueden ser Logs que es texto, donde se pueden definir reglas para ese archivo que no tiene estructura. En esta clasificación de datos la información esta aglomerada y desordenada y no esta almacenada en una base de datos tradicional.

*"**Fragmento de En plena Noche (Antonin Artaud)***
La imaginación, el sueño, toda esta intensa liberación del inconsciente que tiene por finalidad hacer aflorar a la superficie del alma lo que habitualmente está escondido, debe necesariamente introducir profundas transformaciones en la escala de las apariencias, en el valor de significación y en el simbolismo de lo creado. El más allá, lo invisible rechaza la realidad. El mundo ya no se sostiene."

Figura 3. Ejemplo de datos no estructurados (Unstructured Data).
Tomado de (Consultoría editorial, 2014)

2.3.3 Datos Semiestructurados

Con estos datos se puede atender archivos que no son tan planos como los Logs, sin embargo, por su formato son muy dinámicos. Puede ser que la aplicación que genera los datos haya evolucionado empiece a aparecer información nueva en los archivos. Estos datos no presentan una estructura bien definida como los estructurados, pero si tienen una organización definida en los metadatos donde están sus objetos y relaciones. La figura 4 muestra el formato de los datos semiestructurados.



```
{
  "marcadores": [
    {
      "latitude": 40.416875,
      "longitude": -3.703308,
      "city": "Madrid",
      "description": "Puerta del Sol"
    },
    {
      "latitude": 40.417438,
      "longitude": -3.693363,
      "city": "Madrid",
      "description": "Paseo del Prado"
    },
    {
      "latitude": 40.407015,
      "longitude": -3.691163,
      "city": "Madrid",
      "description": "Estación de
        Atocha"
    }
  ]
}
```


Figura 4. Ejemplo de datos semiestructurados (Semistructured Data).
Tomado de: Diego Calvo, Data Scientist.

2.4 Bases de Datos

Una base de datos es un conjunto de datos que están organizados y que tienen una relación entre sí.

2.4.1 Gestores de Bases de Datos

Los gestores de Bases de Datos no son más que un conjunto de programas que permiten trabajar con las bases de datos. Con estos programas se puede tener las siguientes funciones: crear, almacenar, modificar nuestras propias bases de datos.

2.4.2 Bases de Datos Relacionales

A una Base de Datos Relacional se la puede definir como una colección de tablas que almacenan información de manera estructurada, son como las hojas de cálculo ya que una tabla tiene filas y columnas. En cada fila se almacena una entidad.

2.4.3 Sistemas de gestores de Bases de Datos

Lenguaje de consulta estructurada (SQL) es un lenguaje para el manejo de una base de datos relacional, se entiendo por base de datos relacional a una que tenga una modelo relacional que se refiere a una relación existente entre entidades o tablas. Se puede realizar todo el mantenimiento de la base de datos, crear tablas, insertar datos, modificar información, eliminar tablas o información y realizar el proceso más importante que es hacer consultas.
Para manejar una base de datos con SQL se utiliza un intermediario que se conoce como Sistema Gestos de Base de Datos (SGDB) como:

- MySQL
- Oracle
- SQL Server de Microsoft
- PostgreSQL

El objetivo de utilizar una base de datos relacional es que esté conectada a una aplicación para manejar la información desde la misma.

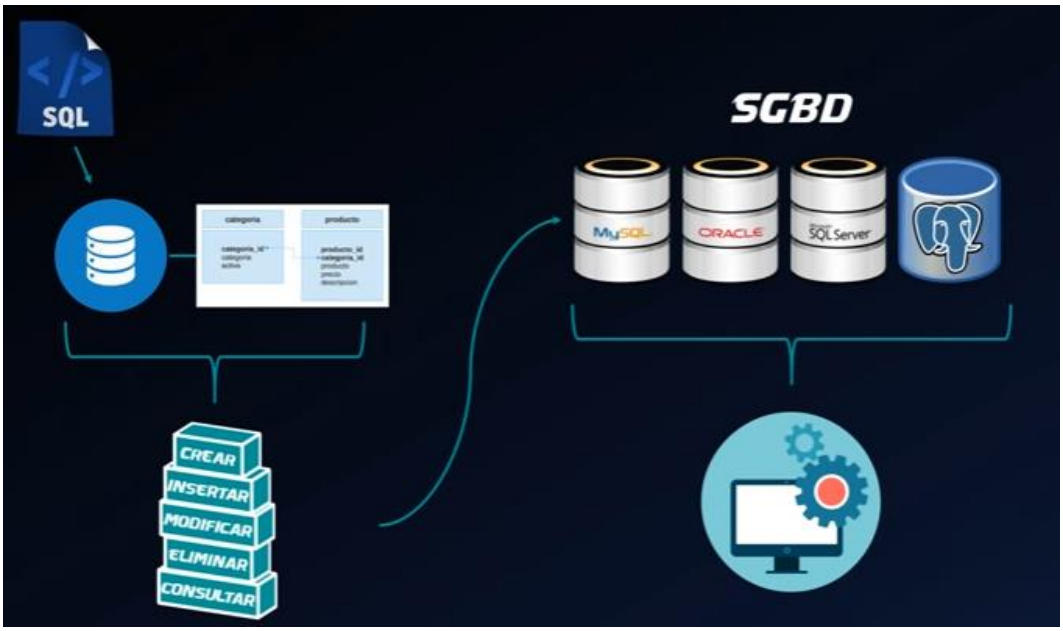


Figura 5. SQL (Structured Query Language).
Tomado de (SQL System, 2016)

2.4.6 Modelos de datos no relacionales

A diferencia de las bases de datos relacionales, estas no poseen un sistema de indetificación que pueda relacionar conjuntos de datos. Además el modelo no relacional permite estructuras más complejas.

2.4.6.1 Estructura del esquema de información

En primer lugar se tienen las bases de datos que almacenan documentos, estas se organizan entorno a la idea de que se debe almacenar un documento autónomo en lugar de filas individuales y columnas definidas como se puede observar en la figura 6.

Figura 6. Ejemplo de esquema anidado de información.

Tomado de (Ingeniería de Big Data, 2017)

2.4.6.2 Guardado en clave y valor

Esta es otra categoria donde no se tiene ningun esquema definido para los datos, todo se guarda como parejas de clave y valor solo se recupera los datos en esquemas de clave y valores como lo muestra la tabla 5.

Tabla 5.

Ejemplo almacenamiento como parejas de clave y valor

CLAVE	VALOR
Nombre1	Pedro
Color	Rojo
Nombre_compania	Microsoft

2.5 Minería de datos

Es una de las fases que se conoce como extracción de conocimiento de bases de datos, incluyen dos tipos de operaciones básicas que además están interrelacionadas que son: el reconocimiento de patrones y la predicción.
Se pretende extraer el conocimiento de los datos distinguiendo la información irrelevante (La y Aprendizaje, 2017).

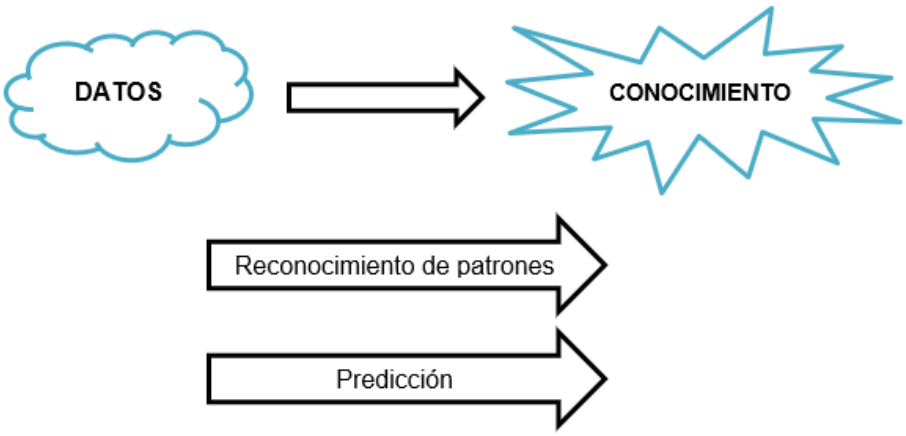


Figura 7. Minería de datos.

Adaptado de (Documental UNED Barbastro, 2018)

El término de minería se utiliza porque la gran cantidad de datos puede considerarse como si fuese una mina y se pretende minar es decir extraer de ella el conocimiento. A la minería de datos también se la conoce con otros nombres como:

- Descubrimiento de conocimiento en bases de datos (KDD)
- Extracción del conocimiento
- Análisis inteligente de datos

2.5.1 Proceso de KDD

En realidad, la minería de datos es una parte de un proceso más amplio llamado proceso de Knowledge Discovery in Data bases (KDD). En la primera

etapa de este proceso se integran los datos en un data warehouse, una vez que los datos han sido integrados se necesita una limpieza adicional. Después de que los datos ya estén acondicionados se puede ejecutar la etapa de minería de datos, consiste en la ejecución de técnicas, cada técnica tiene sus particularidades por ejemplo existen técnicas que trabajan solo con datos discretos o técnicas que trabajan solo con datos continuos.

Una vez obtenidos los datos preparados ya se puede hacer minería de datos sobre ellos y se ejecutan los algoritmos sobre ellos para después obtener modelos, estas son representaciones simbólicas que pueden ser en cierta manera interpretables que representan de alguna manera los datos de origen. Si la interpretación y la evaluación del modelo son favorables se puede decir que se ha encontrado el conocimiento tal como se demuestra en la figura 7 de minería de datos siguiendo el proceso KDD.

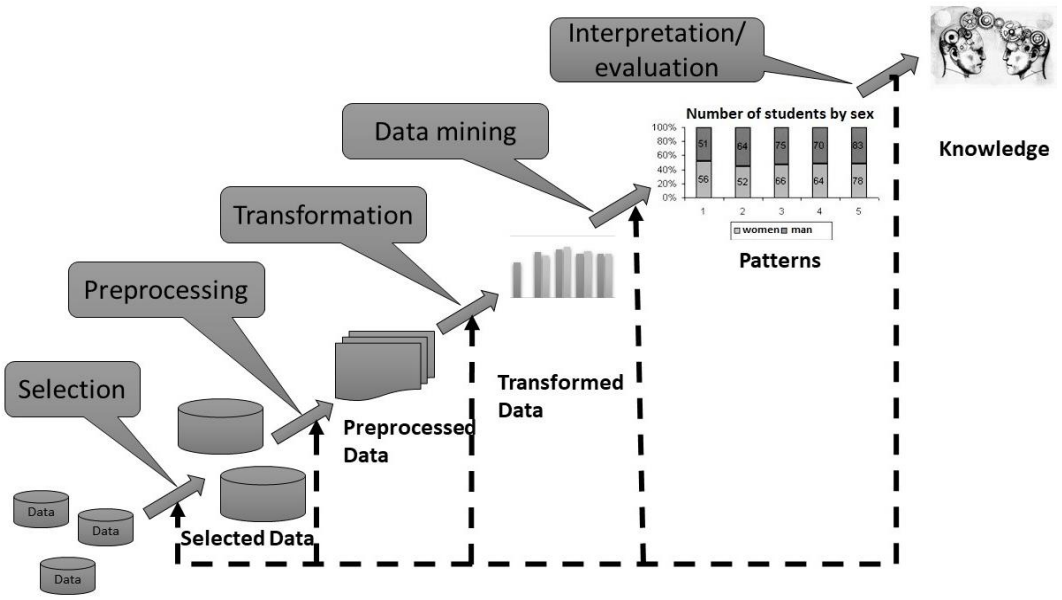


Figura 8. Minería de datos – proceso KDD.

Tomado de (Universidad de Madrid, 2018)

2.5.2 Técnicas de minería de datos

Principalmente la clasificación de las técnicas de minería de datos está descrita por tres modelos que son: técnicas predictivas, técnicas descriptivas y técnicas auxiliares.

2.5.2.1 Técnicas predictivas

En esta técnica es necesario un conocimiento teórico anterior para poder especificar un modelo. Para que este modelo sea aceptado como válido primero se realiza el proceso de minería de datos, después se someterá a prueba y de esa manera comprobar que el valor y la exactitud deseable.

2.5.2.2 Técnicas descriptivas

Es una técnica neutral ya que no se supone algún modelo previo para los datos. Para que se asigne y se cree un modelo se requiere tener el reconocimiento de patrones. Las técnicas de clustering y segmentación están incluidas en este grupo (Datos, Román, García, Jesús y Rueda, n.d.).

2.6 Internet of Things (IoT)

IoT es la tecnología impulsa la prosperidad global, en esencia el Internet de las cosas trata de medir y controlar grandes volúmenes de datos generados por dispositivos de comunicación. Cualquier objeto que sea una fuente de datos está presente en la red, por lo tanto, la conectividad en la red es la base del IoT.

Este concepto de IoT está relacionado con la Ubicuidad. En términos más técnicos se puede decir que consiste en integrar dispositivos electrónicos para conectarlos a Internet, estos deben cumplir requisitos para ser considerados objetos inteligentes. El primero son los componentes computacionales que permiten procesar información como los microcontroladores. Después se

encuentran los sensores que obtienen datos de dispositivos físicos que están en su entorno y convertirla en información procesable como la luminosidad, temperatura y movimiento. Uno de los campos con gran relevancia es la sostenibilidad en un Smart Campus que apunta al diseño y uso eficiente de sistemas de control de presencia como calefacción y alumbrado por zonificación.

Los artefactos que operen en un campus deben programarse para que desarrollen cierta autonomía cuando la energía provenga de fuentes renovables. La implementación de termostatos inteligentes para que aprendan las preferencias y horarios de las personas con el objetivo de hacer uso eficiente de las instalaciones y lograr acercar el mundo físico al mundo digital (Villegas-ch, Palacios-pacheco y Luján-mora, 2018).

2.7 Smart Campus

Las innovaciones en tecnología y el cambio en el comportamiento de los estudiantes brindan nuevas oportunidades a las instituciones educativas como universidades para mejoras los resultados de aprendizaje, mejorar la seguridad en el campus, respaldar las demandas de sus redes.

Las tecnologías implantadas apoyan la resolución de problemas, enfocadas a la buena gestión de recursos medioambientales, la reducción del consumo energético, etc. Un campus inteligente se asemeja a una ciudad inteligente en la emisión de datos, pero enfocado a un entorno universitario. La generación de datos emitidos por sensores en entornos sociales, apoyados con la automatización de los procesos, para transformar esos datos en información.

En un campus universitario se generan miles de datos a diario por la alta concurrencia de estudiantes y personal que trabajan en el mismo. Los campus universitarios se comunican con las ciudades en las que se encuentran en temas tangibles relacionados con la infraestructura y en temas intangibles, como las relaciones sociales o la innovación (Popoola. 2018).

2.7.1 Aplicación del Internet de las cosas y Cloud Computing

La computación en la nube es una arquitectura para proporcionar servicios para las tecnologías de información y la comunicación.

2.7.1.1 Internet de las cosas en la educación

En el desarrollo de la tecnología el Internet de las cosas (IoT) es una parte fundamental para obtener información enviada por sensores, RFID y tecnología de posicionamiento en tiempo real. Esta tecnología aplicada a la educación se maneja por capas, la primera es una capa que percibe los datos de los equipos. Luego sigue la capa de red, que se responsabiliza de la transmisión de datos que vienen de la capa inferior. Finalmente, la capa de aplicación es decir el IoT realiza el trabajo de procesamiento de información.

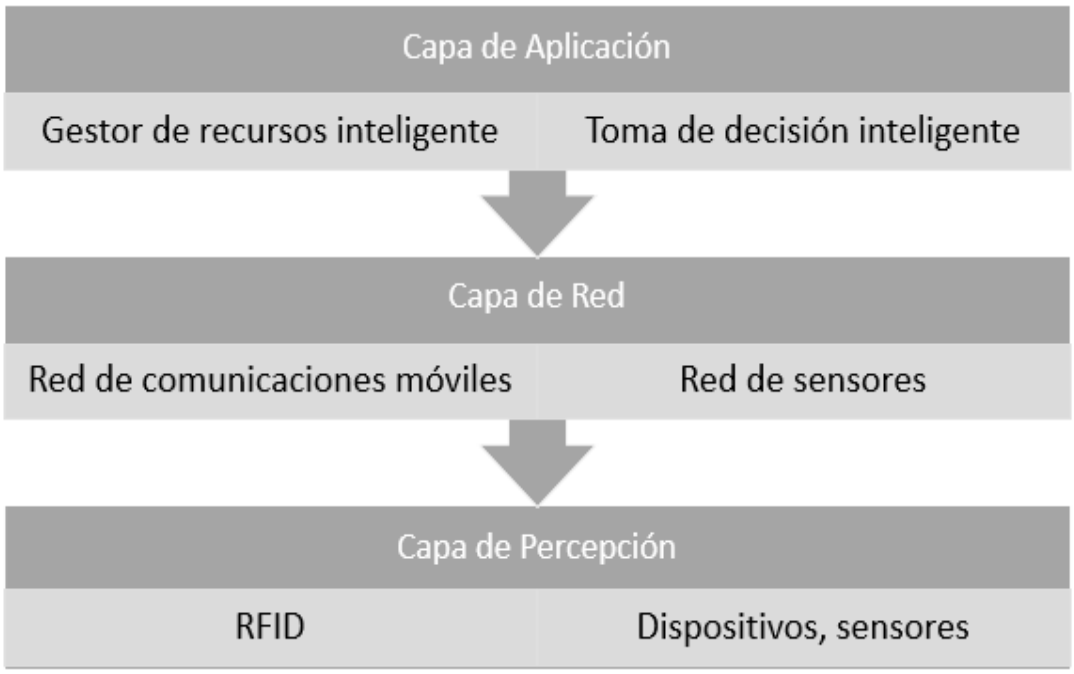


Figura 9. Sistema bajo el entorno IoT.

Adaptado de (College of Engineering, 2016)

2.7.1.2 Cloud computing e Internet de las cosas

En esta etapa se integra la computación en la nube y las plataformas de software. Mediante esta implementación existe la transición de un centro de datos común a la virtualización de servidores.

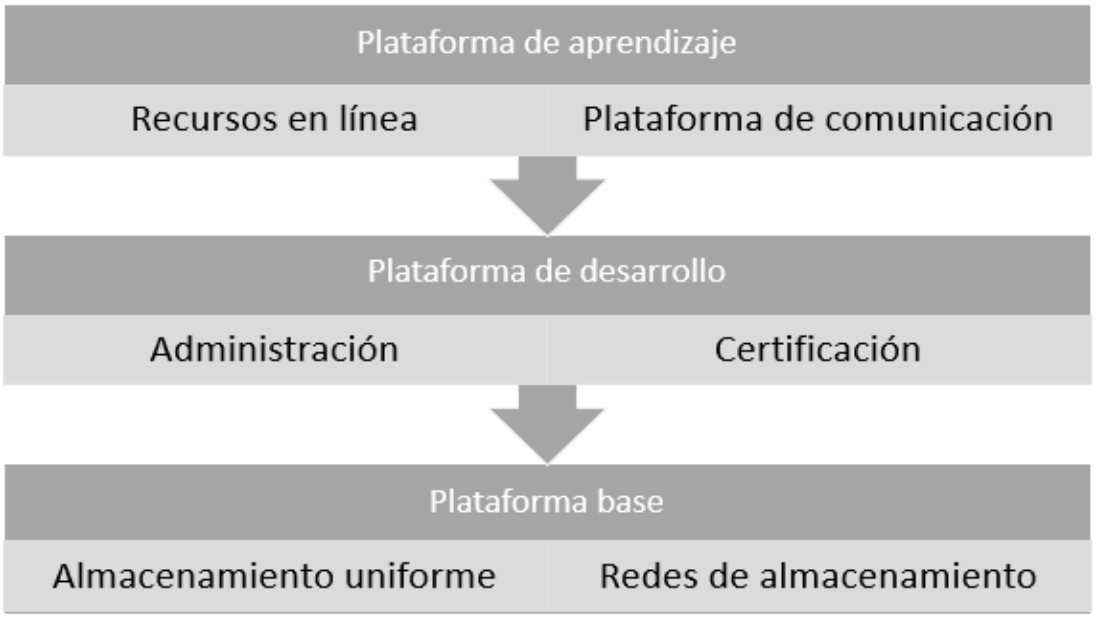


Figura 10. Plataforma del sistema de educación en la nube.
Adaptado de (College of Engineering, 2016)

La realización de esta investigación comenzó con una investigación de bases cientificas como Springer, IEEE, Sensors, enfocadas en IoT, Smart Cities, Smart Campus.

Según Educause, la mayoría de los estudiantes intentan conectar al menos dos dispositivos a la red del campus al mismo tiempo. La conectividad del campus depende de tener una capa física versátil con flexibilidad de red que debe ser planificada, considerar el soporte de múltiples aplicaciones.

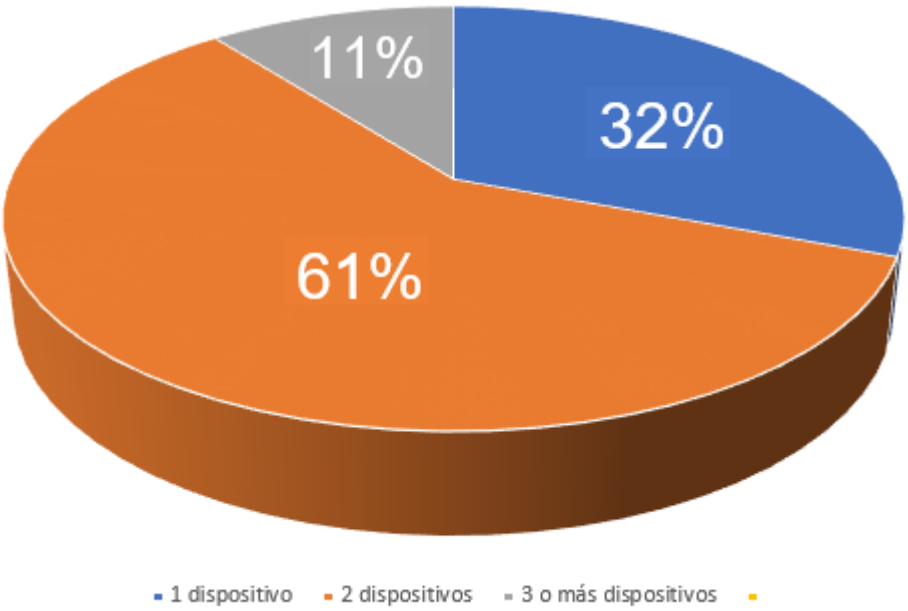


Figura 11. Rendimiento de la red con dispositivos conectados.
Tomado de (Educause, 2018)

Para que un campus universitario llegue a ser considerado un campus inteligente, este ha tenido que pasar por tres etapas: campus tradicional, e-campus y campus digital. Cabe recalcar que un campus digital es un estado antes de convertirse en un campus inteligente. La diferencia que existe entre un campus digital y un campus inteligente se explica en la siguiente tabla.

Tabla 6.

Diferencias entre un campus digital y un campus inteligente

	Campus Digital	Campus Inteligente
Marco técnico	Red de área local	Internet de las cosas, cloud computing, red inalámbrica.
Sistemas de gestión	Sistema aislado	Sistema compartido
Aplicación	Recursos académicos digitales.	Sistema inteligente controlado, interoperabilidad.

Figura 12. Campus Internet de las cosas IoT.
Tomado de (Anixter, 2017)

2.7.2 Características de un Campus Inteligente

El sistema de un campus inteligente integra dispositivos colocados, almacenamineto en la nube. El modelo de una campus se asemeja a un ciudad pequeña que se puede tomar como referencia para ciertas pruebas técnicas y rasgos característicos, como por ejemplo:

Complejidad

Se puede determinar como complejidad a algunos procesos sensibles que realiza un campus universitario.

Diversidad

La diversidad de un campus puede variar por su ubicación geográfica, oferta académica, infraestructura,etc. Cuanto más varien las funciones que presenta un campus el numero de agentes controladores debe ser mayor.

Incertidumbre

Este punto es muy importante ya que se enfrenta a las limitaciones que aún tiene un campus y de prevenir soluciones para entornos cambiantes para que el campus posea un futuro de 10 a 15 años.

La estructura de un campus inteligente debe ser flexible, escalable y evolutiva, donde los procesos se evalúan constantemente y el nivel de reacción ante eventos externos es eficiente (Alvarez-Campana, López, Vázquez, Villagrá y Berrocal, 2017)

2.7.3 Componentes de un Smart Campus

El desarrollo de este artículo está enfocado a la investigación del continuo avance de tecnología de TI, mediante la implementación de IoT, para la construcción de un campus inteligente promoviendo el desarrollo sostenible.
Todos los componentes que intervienen en un sistema que pasan por una serie de procesos interconectados y vinculados para generar un campus inteligente, cuyo objetivo es mejorar la calidad de vida de la comunidad, aplicando TI intensivamente bajo el principio de servicio y sostenibilidad.

El un campus inteligente no se basa en recopilar datos, sino de mejorar el entorno y crear espacios más saludables. Un campus inteligente es una combinación de Internet de las cosas con la computación en la nube.

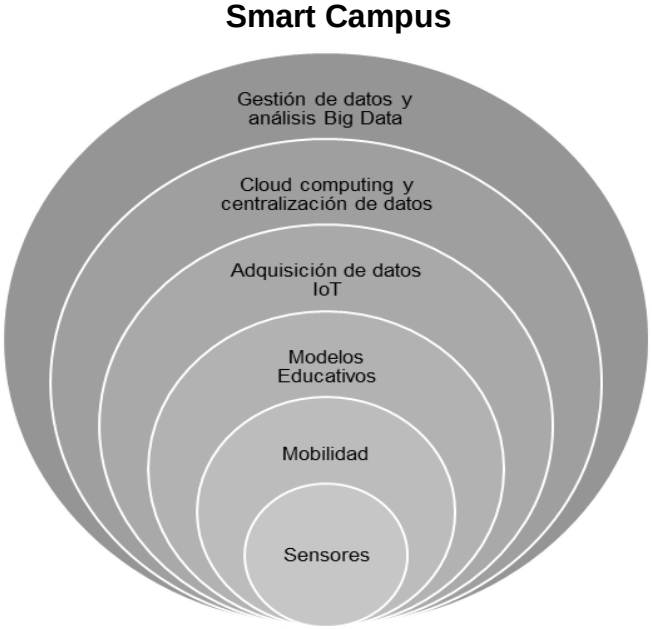


Figura 13. Componentes considerados para un Smart Campus.
Adaptado de: Sensors Artículo Big Data.

3. Tecnologías y Plataformas de Big Data

En esta sección se presentan las tecnologías Hadoop y Spark que se van a utilizar, se podrá conocer las ventajas de virtualización que brindan. En esta sección se van a evaluar las características entre las dos plataformas.

3.1 Apache Hadoop

Figura 14. Logotipo Hadoop.

Tomado de (Apache Software Foundation, 2019)

Es una plataforma que está desarrollado en Java que permite el procesamiento distribuido de grandes cantidades de datos utilizando modelos de programación simples sobre un clúster. (IBM, 2017).

Tabla 7.

Principales características de Hadoop

CARACTERÍSTICAS	
Desarrollador	Apache Software Foundation
Sistema Operativo	Windows, Linux, OS X
Última versión	Versión 3.1.2
Licencia	Apache 2.0
Plataforma	Java
Lenguaje de programación	Java

Hadoop es una framework que permite usar computadoras genéricas para procesar datos de forma distribuida. Esto quiere decir que así se tenga un hardware muy limitado se puede instalar Hadoop, ya que se puede tener muchas máquinas de bajo costo o de uso genérico, entonces Hadoop se ha posesionado como la herramienta básica para hacer Big Data por tener la característica de poder procesar de forma distribuida y estar sobre la máquina virtual de Java.

Tabla 8.

Mezcla de modelos distribuidos para la creación de Hadoop

Distributed Storage Model	Distributed Compute Model
• Google File System	• MapReduce
Almacenamiento datos en grupos masivos de máquinas económicas. Tolera fallo de hardware.	Enviar datos para procesarlos. Simplifica la programación distribuida.

3.1.1 Características básicas

Procesamiento distribuido

La idea de Hadoop es poder distribuir los datos de forma que cada nodo del clúster de máquinas procese una parte de los datos, de esta manera se gana velocidad.

Eficiente

Consigue procesar los datos en poco tiempo.

Económico

Es un framework económico porque se puede escalar fácilmente de manera horizontal.

Fácilmente escalable

Si el sistema se queda pequeño únicamente se debe añadir un nuevo nodo que se lo añade a un clúster de máquinas.

Tolerante a fallos

Usa la alta disponibilidad y además usa la replicación, los datos suelen estar replicados con replicación tres en el HDFS que es el sistema de almacenamiento de Hadoop de forma que si un nodo cae se tienen la notificación en el resto de nodos.

Open source

Es un proyecto de código abierto.

3.1.1.1 Cuando usar Hadoop

Apache Hadoop es una tecnología de código abierto, escrito en java que permite el procesamiento distribuido de grandes cantidades de datos en grupos de computadoras usando modelos de programación simples. Proporciona un almacenamiento de datos escalable, eficiente con prevención para su crecimiento a futuro. Hadoop es muy conocido y usado por compañías para buscar y procesar registros, análisis, cualquier tipo de dato. Su peculiaridad es manejar distintos tipos de datos, entonces puede trabajar con datos estructurados, semiestructurados y no estructurados.

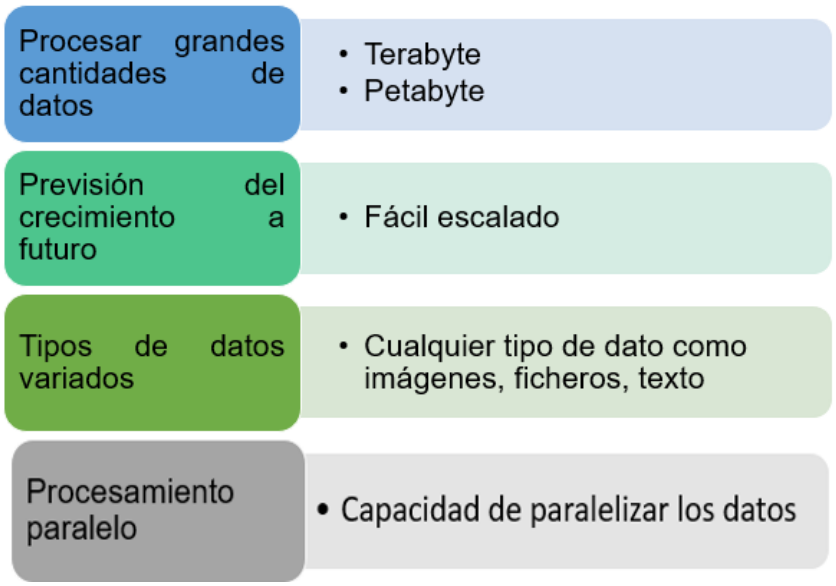


Figura 15. Uso Hadoop.

Adaptado de (OpenWebniars, 2018)

3.1.1.2 Cuando no usar Hadoop

Apache Hadoop tiene una posible limitación en los tiempos de ejecución, es que Hadoop necesita crear múltiples objetos y el intercambio de datos en MapReduce es lento debido a la replicación. Cuando se quiere modificar los datos, solamente se puede añadir contenido al final o eliminarlo. Su rendimiento es afectado en relación al tamaño de la ventana y el tiempo de ejecución.

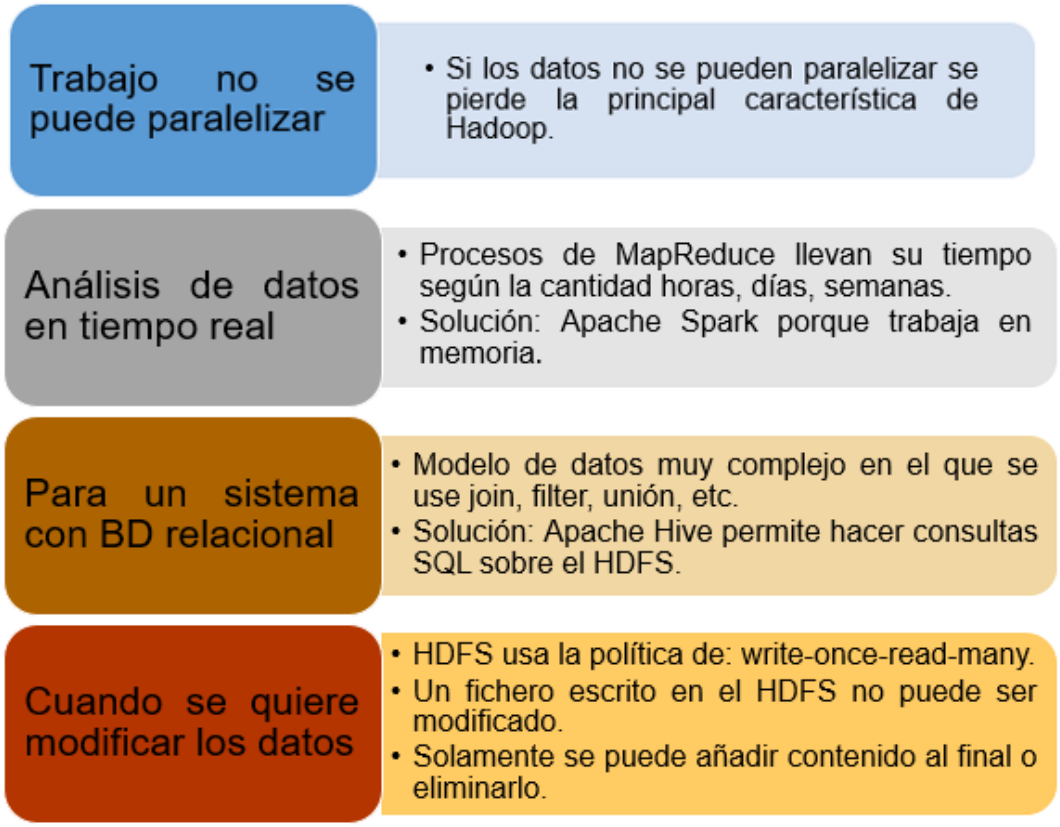


Figura 16. No uso Hadoop.
Adaptado de (OpenWebniars, 2018)

3.1.2 Arquitectura básica, módulos de Hadoop

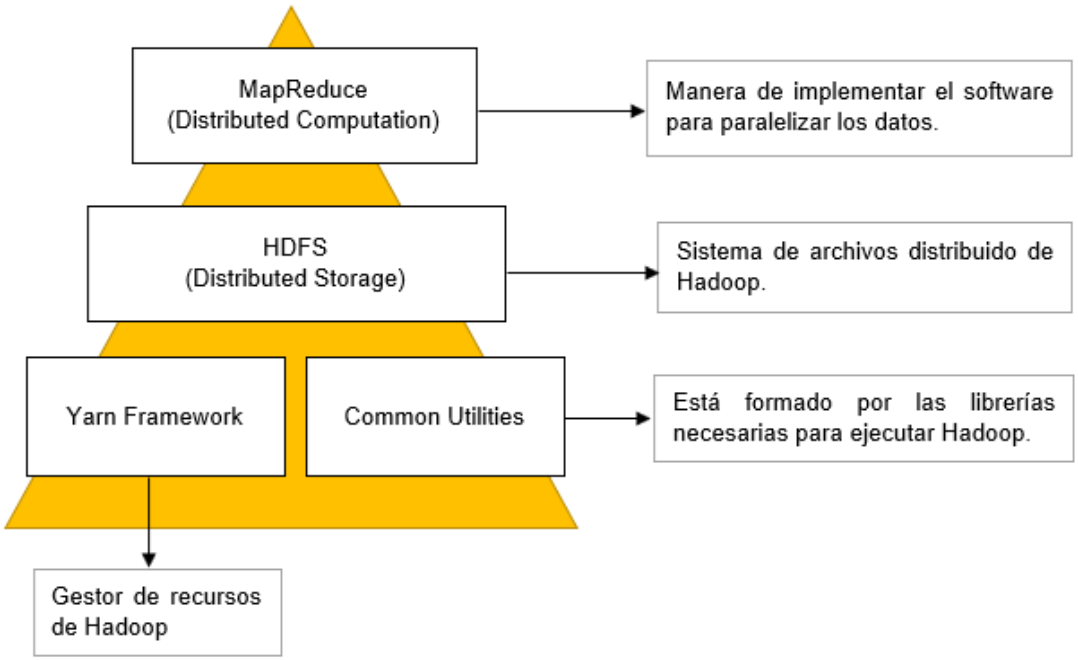


Figura 17. Arquitectura de Hadoop.
Adaptado de (OpenWebniars, 2018)

3.1.3 Proceso de Hadoop

Ya sea que venga de una colección estática o de un flujo de datos mediante este proceso se puede generar almacenamiento tradicional, luego se procesa y se analiza con el concepto de MapReduce. Puede existir una segunda fase de procesamiento donde se refina el resultado. Finalmente se genera el resultado. El sistema distribuido de archivos de Hadoop es HDFS, diseñado para almacenar archivos grandes.

MapReduce es una técnica de procesamiento y un modelo de programación para computación distribuida. La principal ventaja de MapReduce es que es más fácil escalar el procesamiento de datos en múltiples nodos de computación.

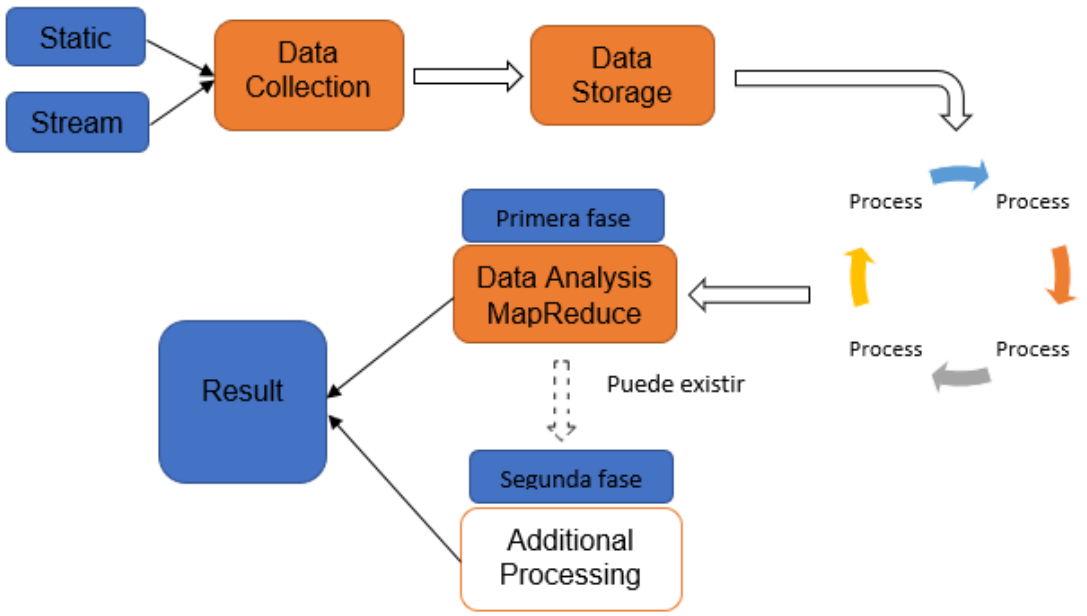


Figura 18. Esquema del proceso de Hadoop.

3.1.4 MapReduce

MapReduce es un modelo de programación orientado a la ejecución paralela y distribuida a través de múltiples computadoras y se utiliza principalmente para trabajar con grandes cantidades de datos de la orden de terabytes o de petabytes (Samadi, Zbakh y Tadonki, 2017).

La idea de MapReduce es otorgar una manera simple, rápida, escalable y resistente a fallos de trabajar con grandes archivos o con grandes fuentes de datos. Como se puede observar en la figura 18 el proceso que realiza MapReduce es que toma un input que puede ser un archivo, una base de datos, etc. Convierte una serie de valores en valores de clave y valor.

Pasa por una función denominada MAP, aquí cada clave y valor se convierte en otra clave y valor. Esta estructura después pasa por un proceso de reducción REDUCE, donde estas claves y valores se combinan para producir una nueva estructura de claves y valores.

Para aprovechar de manera eficiente el procesamiento en paralelo que brinda Hadoop se debe realizar la consulta como MapReduce, primero se debe hacer pruebas locales sencillas para luego ejecutarlo en un clúster de máquinas.

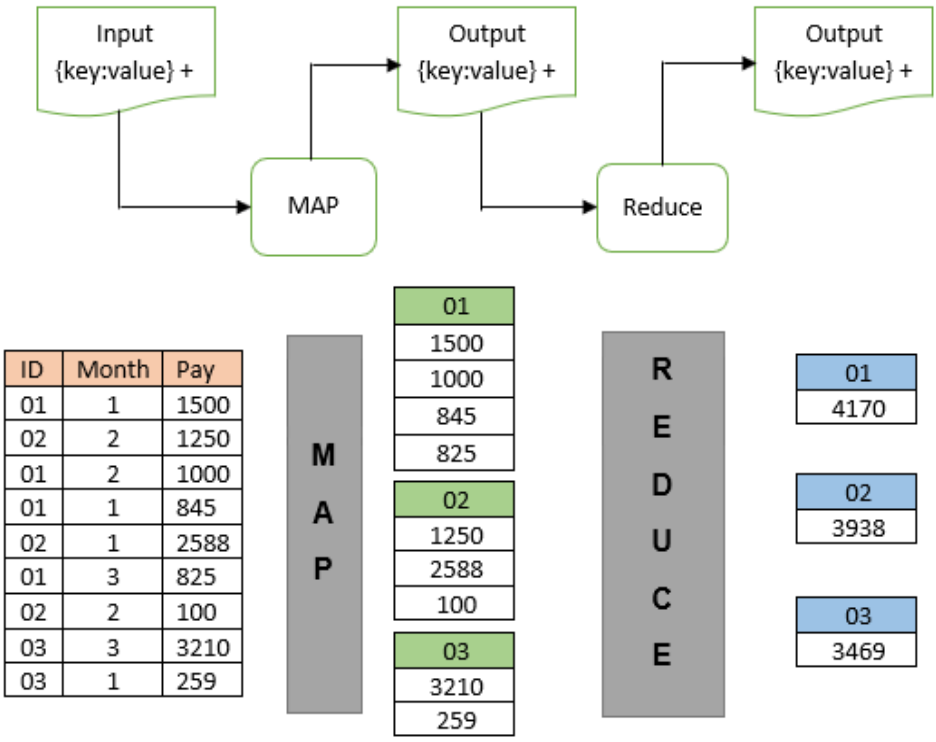


Figura 19. Funcionamiento de MapReduce.
Adaptado de (SG Campus, 2017)

3.1.5 Analizar datos con Hadoop

3.1.5.1 Java MapReduce

Una vez realizado el análisis del funcionamiento de MapReduce, se lo tiene que expresar en código para lo cual se necesitan tres aspectos:

- Una función map
- Una función reduce
- Código para ejecutar el trabajo

3.1.5.2 Prueba de funcionamiento

Se debe probar un trabajo MapReduce con un pequeño conjunto de datos para eliminar cualquier problema existente en el código.

Para hacer esta prueba se debe instalar Hadoop en modo independiente, en este modo se ejecutará Hadoop utilizando el sistema de archivos local con u corredor de trabajos local.

3.1.4.2 Escalamiento

Para escalar los datos serán almacenados en un sistema de archivos distribuido (HDFS), para que Hadoop pueda poner en cada máquina el cálculo que realizó MapReduce.

Flujo de datos

Teniendo en cuenta que un trabajo de MapReduce se refiere a un trabajo que el cliente desea realizar que son los datos de entrada. Cuando Hadoop ejecuta el trabajo dividiéndolo en tareas que son: asignar tareas y reducir tareas.

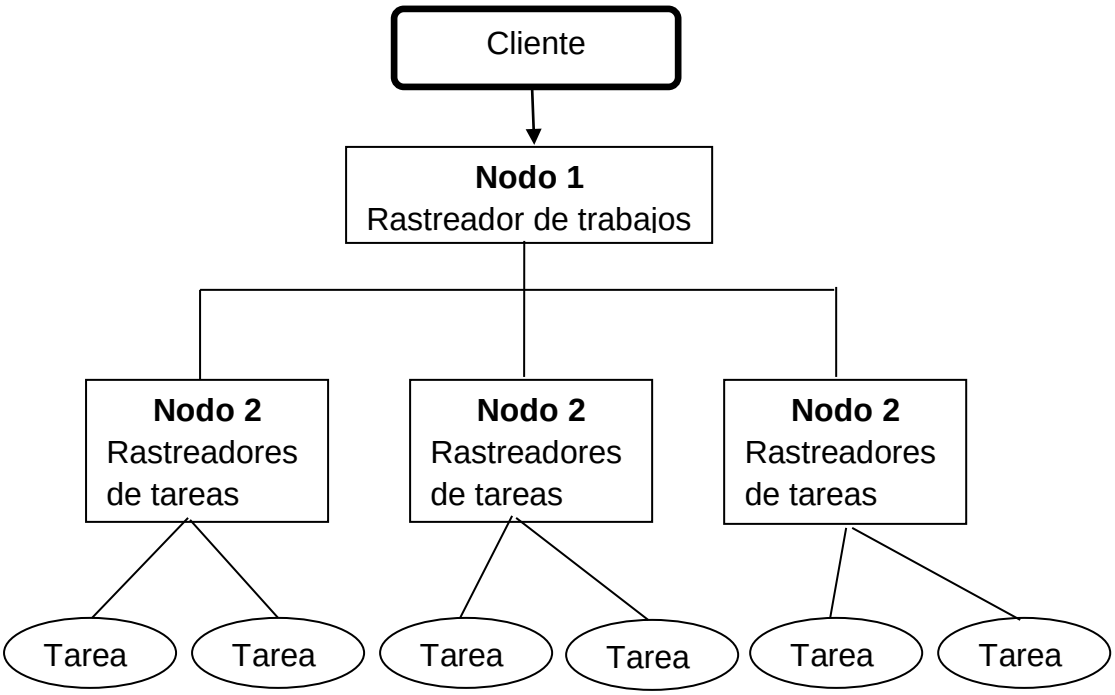


Figura 20. Proceso de ejecución del trabajo.

Existen dos tipos de nodos que controlan la ejecución del trabajo estos son el rastreador de trabajos (jobtracker) y rastreador de tareas (tasktracker).
El jobtracker es el encargado de coordinar los trabajos que se están ejecutando en el sistema, específicamente trabajan en las tareas para que vayan a los tasktrackers y se ejecuten ahí. En este punto los tasktracker ejecutan las tareas y constantemente envían informes de progreso al jobtracker, que este va haciendo un registro de todos los informes del trabajo. Si una tarea llegara a fallar, el jobtracker puede reprogramar esta tarea en un tasktracker diferente.

3.1.5 HDFS

Sistema distribuido de archivos o Storage de Hadoop, donde se pueden almacenar datos estructurados y no estructurados. (Dhruba Borthakur, 2008).
Sistema de archivos llamado HDFS, todo lo que se vaya hacer en Hadoop se pone en HDFS. El Sistema de archivos distribuidos de Hadoop (HDFS) es un sistema de archivos distribuidos que puede almacenar datos en múltiples servidores, con paralelismo de acceso y tolerancia a fallas. A diferencia de otros sistemas de archivos distribuidos, HDFS está diseñado para ser construido a partir de un componente de productos de bajo costo que requiere que sea altamente tolerante a fallos.
Usar el modelo Master / Slaves es decir que existe una máquina que es el maestro y los demás son los esclavos de esa máquina.
Se ejecuta sobre el sistema de archivos de la máquina, pero es importante destacar que no son lo mismo. El HDFS sabe dónde este cada dato, sabe que tamaño tienen los archivos y sabe los nombres de los mismos, también los permisos que tiene cada usuario sobre cualquiera de esos archivos. Para conocer todos los registros internos y características de los archivos y datos, HDFS divide los datos en bloque de igual tamaño y va distribuyendo por los nodos que son parte del clúster.

Archivos muy grandes, se refiere a los archivos que contienen niveles de información muy grandes como megabytes, gigabytes o terabytes.

Acceso a datos de transmisión, HDFS se basa en que el patrón de procesamiento de datos es el de lectura una vez y lectura muchas veces.
Hardware básico, quiere decir que Hadoop no necesita de equipos de hardware muy costosos para ejecutarse ya que está diseñado para que pueda ser ejecutados en clústeres de hardware básico.

Acceso a datos de baja latencia

HDFS este hecho para entregar alto rendimiento por lo que se tiene aplicaciones que requieren baja latencia a los datos entonces no funcionara bien con HDFS.

Muchos archivos pequeños

Como el NameNode es el que guarda los metadatos de los archivos en la memoria, el límite es la cantidad de memoria que tiene el NameNode. Si cada archivo, directorio y bloque tiene 150 bytes y tuviera 1 millón de archivos se necesitaría 300 MB de memoria aproximadamente.

3.1.6 Componentes de HDFS

3.1.6.1 Bloques

Un disco tiene un tamaño de bloque, que esto representa la cantidad mínima de datos que puede leer o escribir.
Los bloques del sistema de archivos suelen tener un tamaño de no muchos kbytes mientras que los bloques de disco tienen 5120 bytes. Los usuarios no perciben esto mientras lee o escribe un archivo.

3.1.6.2 Flujo de datos

Para tener una idea de cómo está formado un archivo y de qué manera fluyen los datos desde el cliente hacia hasta el HDFS, cual es el rol del NameNode y del DataNode.

3.1.6.3 Proceso de un archivo leído

Paso uno, el cliente abre el archivo que desea leer, este proceso se realiza llamando a open () en el objeto de sistema distribuido de archivos.
Paso dos, el objeto de sistema distribuido de archivos llama al NameNode para determinar las ubicaciones de los bloques.
Paso tres, el cliente llama a read () mientras se transmite.
Paso cuatro, se transmiten los datos desde el DataNode hasta el cliente, llama a read ().
Paso cinco, cuando se llega al final del bloque, el flujo de entrada de datos cierra la conexión al DataNode, luego buscar el mejor DataNode para el siguiente bloque.
Paso seis, el flujo de entrada de datos hace que los bloques se lean en orden, también se llamará al NameNode para obtener las ubicaciones de los datos del nodo que servirán al siguiente grupo de bloques.

3.1.6.4 Proceso de un archivo escrito

Paso uno, en el sistema de archivos distribuido con el comando create (), el cliente crea el archivo.
Paso dos, el sistema de archivos distribuido hace una llamada al NameNode para crear un nuevo archivo.
Paso tres, mientras el cliente va escribiendo datos, el flujo de salida de datos los divide en paquetes.
Paso cuatro, el paquete es reenviado desde el segundo DataNode donde fue almacenado hacia el tercer DataNode.
Paso cinco, un paquete puede ser eliminado de la cola siempre y cuando este haya sido reconocido por los DataNodes en el camino.

Paso seis, se utiliza cuando el cliente ya ha terminado de escribir datos, esto se realiza llamando al comando close ().

Paso siete, se vacían los paquetes que sobren en el canal que conduce hacia el DataNode, este contacta al DataNode. y le indica que el archivo está completo.

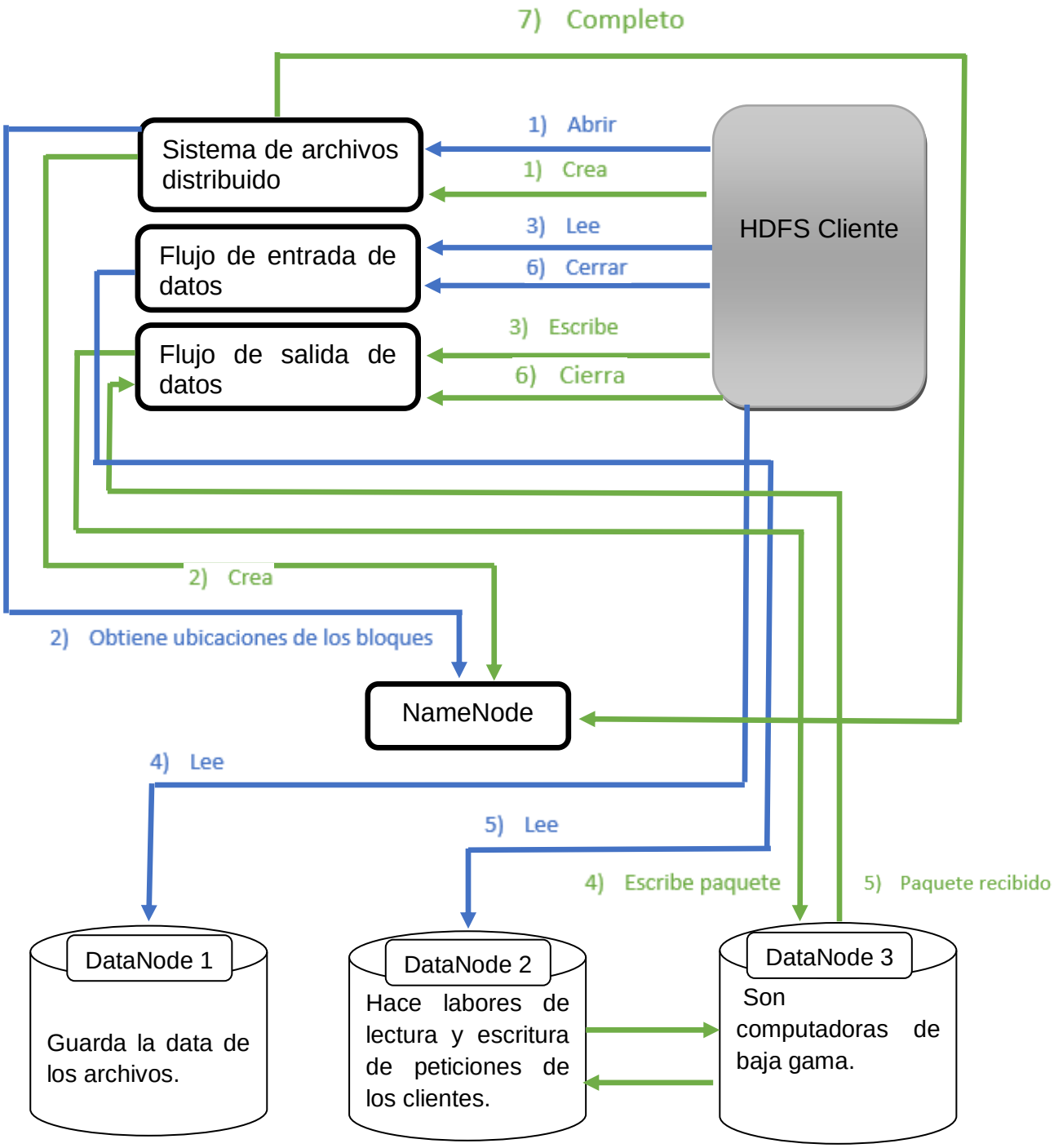


Figura 21. Diagrama de procesos de flujo de datos.

Un clúster HDFS tiene dos tipos de nodos que son maestro y trabajador. Se tiene un solo NameNode (maestro) y varios DataNodes (trabajadores).
El NameNode es el encargado de gestionar el espacio de nombres del sistema de archivos. Mantiene los metadatos, archivos y directorio en el árbol del sistema. Esta información generada se va guardando de manera constante en el disco local. Los DataNodes son una herramienta que tiene el sistema de archivos su función es almacenar y recuperar bloque de archivos. También van informando constantemente al NameNode sobre los bloques que se van almacenando. Hadoop ofrece dos mecanismos para hacer al NameNode resistente a fallos.

La primera solución, es realizar una copia de seguridad de los archivos que forman parte del estado persistente de los metadatos.
La segunda solución, es ejecutar una NameNode secundario, que sirve como un respaldo, pero a pesar de ser llamado NameNode no actúa como tal. Su función es combinar la imagen del espacio de nombres con el registro de edición para evitar que el registro de edición sea muy grande. Este segundo NameNode se ejecuta en una maquina física separada.

HDFS de alta disponibilidad

La replicación de NameNode y el uso del NameNode secundario para crear punto de control protege contra la perdida de datos, pero no proporciona una alta disponibilidad del sistema de archivos.
Como el NameNode es el único repositorio de datos y asignación de archivos es un punto crítico en las fallas, porque si falla, todos los clientes incluido los trabajos de MapReduce no podrían leer ni escribir.

Sistemas de archivos Hadoop
Hadoop tienen una noción imprecisa de sistemas de archivos por lo que HDFS es solo una implementación. Existen varias implementaciones como se describe en la Tabla 9.

Tabla 9.

Sistema de archivos Hadoop

Sistema de archivos	Descripción
Local	Sistema de archivos para un disco conectado localmente.
HDFS	HDFS funciona bien junto a MapReduce.
HFTP	Da acceso solo de lectura a HDFS sobre HTTP
HSFTP	Da acceso solo de lectura a HDFS por medio de HTTPS.
WebHDFS	Da acceso seguro de lectura y escritura a HDFS por medio de HTTPS y HSFTP.
HAR	Consiste en un sistema de archivos dividido en capas sobre otro sistema de archivos usado para archivar y reducir el uso de la memoria NameNode.
KFS	Es un sistema de archivos como HDFS o GFS escrito en C++.
FTP	Sistema de archivos respaldado por un servidor FTP.
S3 nativa	Sistema de archivos respaldado por Amazon S3.
S3 bloques	Sistema de archivos respaldado por Amazon S3 que almacena los archivos en bloques.
RAID distribuido	Versión RAID del HDFS para almacenar archivos.
View	Tabla de acoplamiento que posee el cliente para otros sistemas de archivos.

3.1.7 YARN

Cuando existen clústeres muy grandes, el sistema de MapReduce se ve afectado y se provocan cuellos de botella. Gracias a este problema Yahoo diseño la próxima generación de MapReduce.

Como ya se explicó anteriormente el problema que tenía MapReduce, Yarn busca resolver las deficiencias de escalabilidad, dividiendo en las responsabilidades del jobtraker en entidades separadas.

Yarn es un componente importante de Hadoop y se lo agrega para proporcionar un mejor rendimiento, el modelo actual ofrece ciertas ventajas en comparación con el clásico.

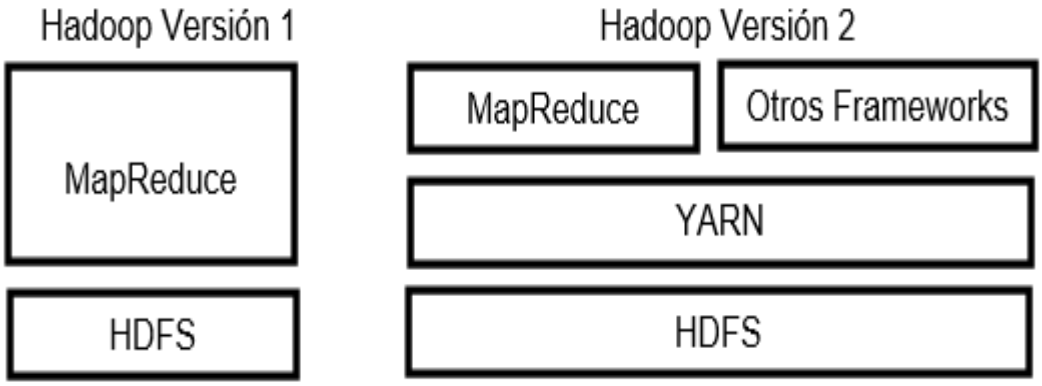


Figura 22. Modelo clásico y actual.

Tomado de (Coso IT, 2018)

Detrás de la versión dos de MapReduce, estas son las limitaciones que poseía MapReduce versión 1. La escalabilidad por el cuello de botella que causa al tener un solo jobtracker. Según Yahoo los limites prácticos de ese diseño se logran teniendo un grupo de 5000 nodos y 40000 tareas que se ejecutan simultáneamente. Los recursos de la máquina de cada nodo esclavo se dividen por un administrador de clúster. Hadoop fue creado solo para implementar trabajos de MapReduce.

3.1.7.1 Arquitectura de YARN

Yarn es la próxima generación de Hadoop que ofrece diversas ventajas en comparación con el clásico motor de MapReduce. Yarn representa a un negociador de recursos, es una capa que separa la capa de administración de recursos y los componentes de procesamiento. MapReduce mueve la administración de recursos (como la infraestructura para monitorear nodos, asignar recursos y programar trabajos) al Yarn.

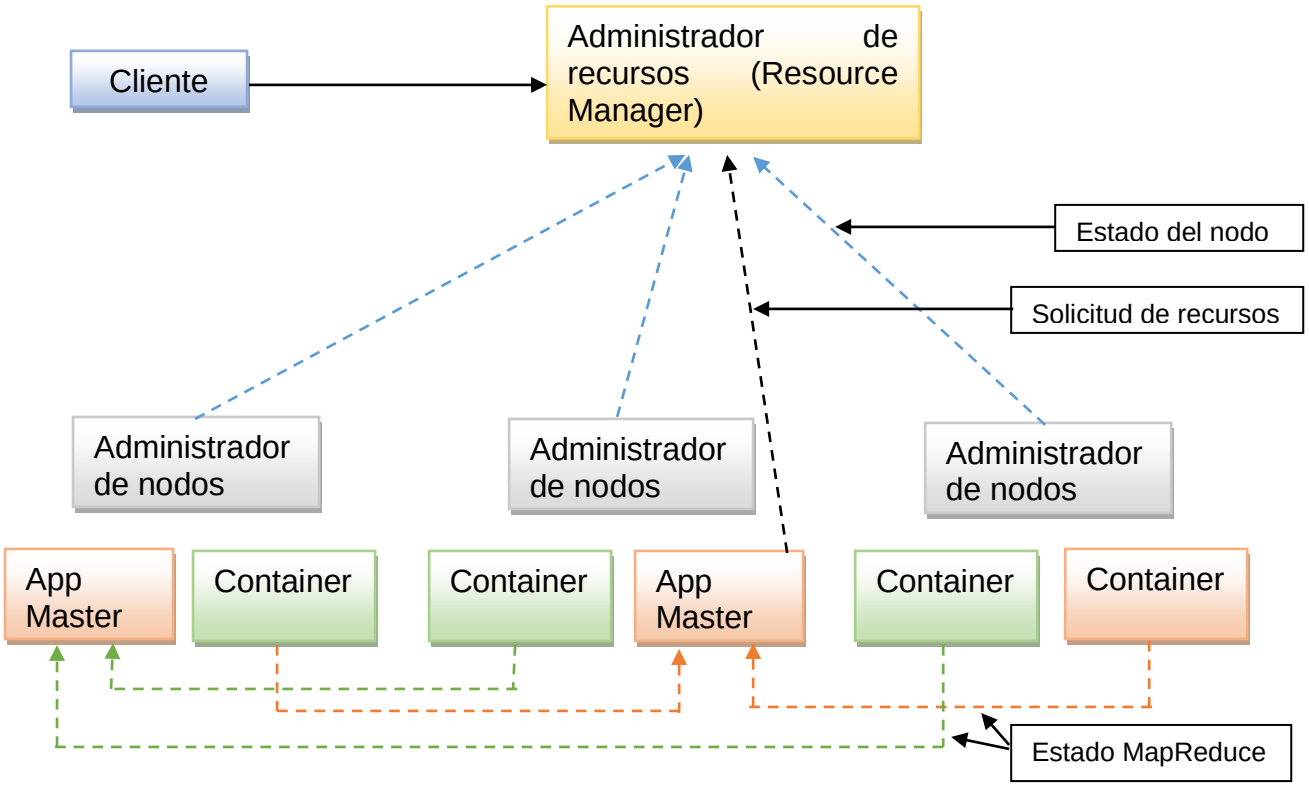


Figura 23. Arquitectura de YARN.
Adaptado de (Edureka, 2018)

3.1.8 Componentes de YARN

3.1.8.1 Administrador de recursos

Es el encargado de gestionar la asignación de recursos en el clúster por lo que es la última autoridad en la asignación de recursos del contenedor. El administrador de recursos posee dos componentes el planificador y el gestor de aplicaciones. Sus recursos son planificados de manera global. El programador es el responsable de asignar recursos a las aplicaciones, pero no ofrece garantías sobre el reinicio de la tarea fallida.

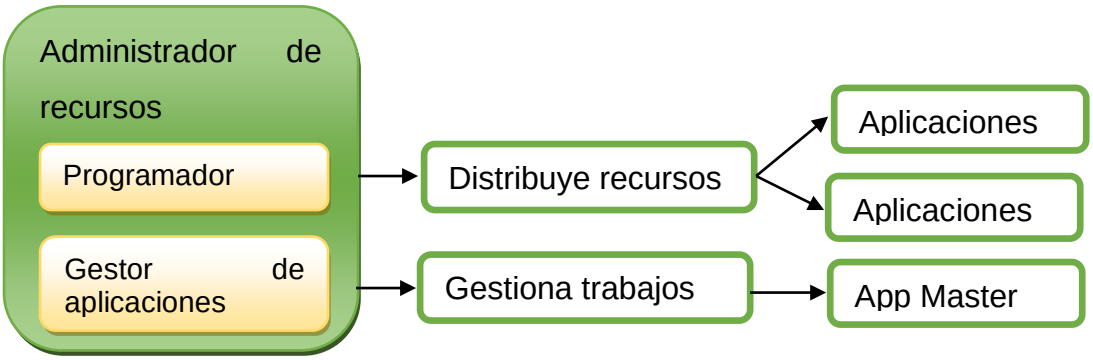


Figura 24. Administrador de recursos.

Tomado de (Coso IT, 2018)

3.1.8.2 Aplicación Master

Es la encargada de gestionar las necesidades de recursos de aplicaciones individuales, es decir negocia con el administrador de recursos y ejecuta procesos específicos de la aplicación.

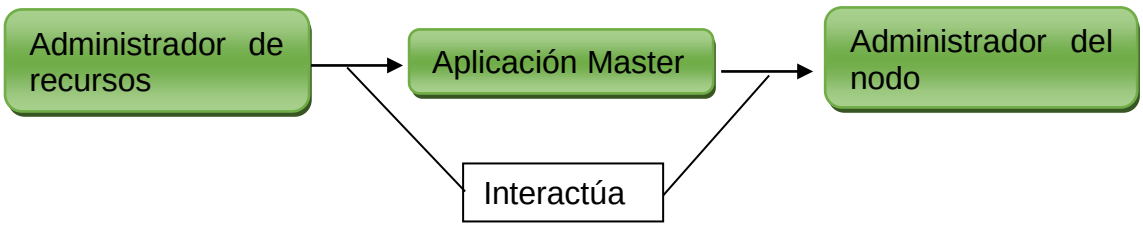


Figura 25. Aplicación Master.

Tomado de (Coso IT, 2018)

3.1.8.3 Administrador de Nodos

Es un rastreador de tareas generalizado, se usa para supervisar a los contenedores que se ejecutan en el nodo del clúster.

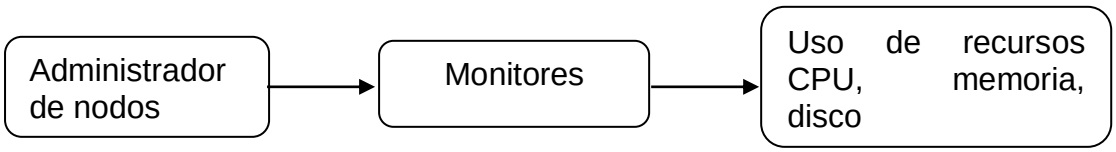


Figura 26. Administrador de nodos.
Tomado de (Coso IT, 2018)

3.1.8.4 Contenedor

Es una colección de recursos físicos, como los núcleos y discos de RAM, CPU en un solo modo. Es el que puede ejecutar diferentes tipos de tareas como matemáticas. Hay múltiples contenedores en un solo nodo. Cada nodo en el sistema tiene múltiples contenedores.

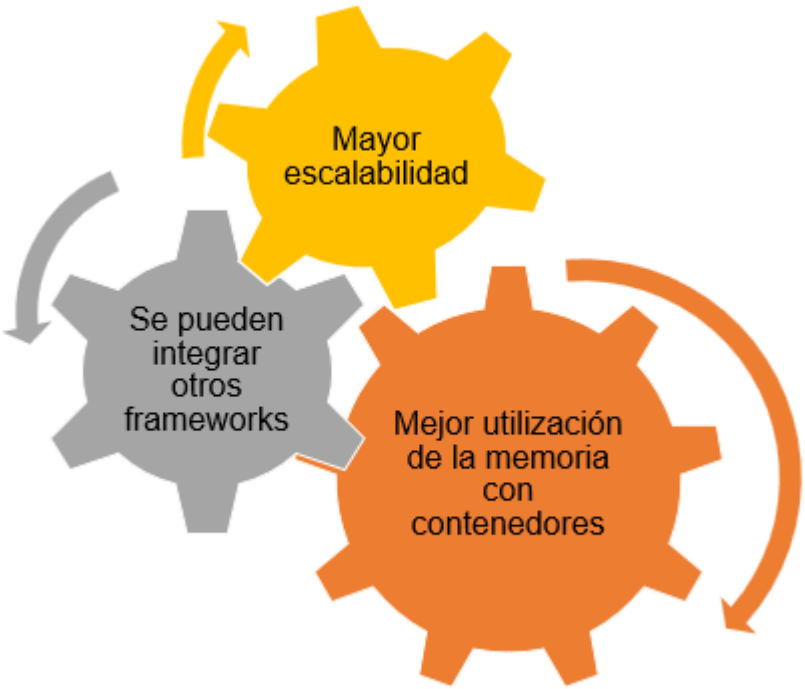


Figura 27. Beneficios de YARN.

3.1.8.5 Ambiente de desarrollo

En la sifuiente imagen se realiza el ambiente que requiere la maquina cliente, en este caso las maquinas itilizadas por los estudiantes y perosnal del campus son las que poseen el ambiente de desaroolo.

3.2 Apache Spark

Figura 28. Logotipo de Apache Spark.

Tomado de (Apache Software Foundation, 2019)

Es un sistema de computación que se basa en Apache Hadoop o en sus procesos MapReduce, principalmente su filosofía es dividir o paralelizar el trabajo ya que normalmente se instala en un clúster de máquinas (OpenWebinars, 2018).

Tabla 10.

Principales características de Spark

CARACTERÍSTICAS	
Desarrollador	UC Berkeley AMPLab, Databricks
Sistema Operativo	Windows, Linux, OS X
Última versión	Versión 2.4.2
Licencia	Apache 2.0
Plataforma	Java
Lenguaje de programación	Java, Scala, Python, R

A partir del año 2009 comenzó la primera versión alfa de Apache Spark. Se sitúa dentro del ecosistema de Apache Hadoop, por lo cual es como familia de Hadoop.

3.2.1 Características principales

La principal característica que posee es que está ligado con Apache Hadoop.
Trabaja en memoria por lo cual se consigue mucha velocidad de procesamiento, es 100 veces más rápido. También permite trabajar en disco, de esa manera si se tiene un fichero muy grande, muchos ficheros o una cantidad de información muy grande que no cabe completamente en memoria esta herramienta permite almacenar parte en disco con lo cual se pierde velocidad, por lo que se intenta encontrar un punto de equilibrio para saber que se almacena en memoria y que en disco.
Proporciona API Java, Scala, Python, R. Permite el procesamiento en tiempo real de los datos. RDD (Resilient Distributed Dataset), todas las transformaciones que se van realizando sobre los RDD no se resuelve sino, se van almacenando en un grafo dirigido y cuando se ejecute una acción, entonces comenzara a ejecutarse la herramienta.

3.2.2 Librerías Externas

Spark tiene librerías que por defecto ya vienen integradas, gracias a esto puede soportar consultas SQL, Machine Learning, Streaming de datos. También maneja proyectos como Thunder que es una librería capaz de analizar datos neuronales a gran escala. (Jean Paul Fannes, 2017).

DAG (Gráfica Acíclica Dirigida) grafo dirigido que no posee ciclos, por cada nodo del grafo no habrá caminos que empiecen y acaben en sí mismo como se puede observar en la figura 13.
Cada tarea de Spark crea un DAG de etapas de trabajo para que sean ejecutadas en un clúster (Marques, 2014).

El motor de DAG trabaja asignando operaciones a los procesadores disponibles.

Según Spark se define como una colección de elementos que es tolerante a fallos y capaz de operar en paralelo. (Abraham Requema, 2018).

Es la principal abstracción de datos que tiene Apache Spark, el tipo de dato básico que contiene Spark.
Los distintos nodos del clúster están particionados, cuando se instala apache Spark se lo hace en un clúster de máquinas por lo cual los RDD se encuentran distribuidos en esas máquinas y consigue la tolerancia a fallos.
Se suelen crear a partir de un fichero del HDFS que es el sistema de archivos distribuidos de Hadoop.
Usa la evolución perezosa, es decir que todas las transformaciones que se vayan haciendo a los RDD se van a ir almacenando en un DAG y no se resolverá hasta que la herramienta este obligado a realizar esas transformaciones.
La evolución perezosa tiene la ventaja de que se gana tiempo y el inconveniente es que si falla no se va a ver hasta que se resuelva el grafo completo.

3.2.4.1 Serialización de RDD

Mediante la aplicación de la serialización se busca que el desempeño de la aplicación distribuida, se presentan problemas cuando los archivos no pueden transmitirse porque están compuestos de una gran cantidad de bits. En busca de la solución de este problema Spark proporciona dos librerías de serialización.

Serialización Java, como se lo usa con un gran volumen de datos, es lento y se lo usa como un parámetro por defecto.

Serialización Kryo, mejora la comunicación de los paquetes más pequeños ya que necesita mejorar la eficiencia.

Transformaciones después de realizar una transformación, se obtiene un nuevo RDD que está fundamentado en el original que fue modificado. Acciones consiste en aplicar cierta operación a un RDD para conseguir un resultado.
Spark RDD es el método que se utiliza para almacenar y procesar creando datos que residen en la memoria denominados RDD que son una colección de objetos clasificados en dataset y dataframes.

Cada RDD tienen las siguientes políticas:

- No se debe alterar, pero se puede crear otro RDD.
- Se lo puede distribuir a todos los nodos del clúster.
- Solo puede residir en memoria.

Existen dos tipos de operaciones de transformación que son los siguientes:

Narrow

Se utiliza cuando se van a tratar los datos y estos están en la misma partición de RDD, no es necesario mezclar los datos como, por ejemplo:

- filter ()
- sample ()
- map ()
- flatMap ()

Wide

Se utiliza cuando se van a tratar los datos y estos están en diferentes particiones de un RDD, si es necesario mezclar los datos como, por ejemplo:

- groupByKey ()
- reduceByKey ()

En la figura 29 está una representación de ambos tipos de transformaciones.

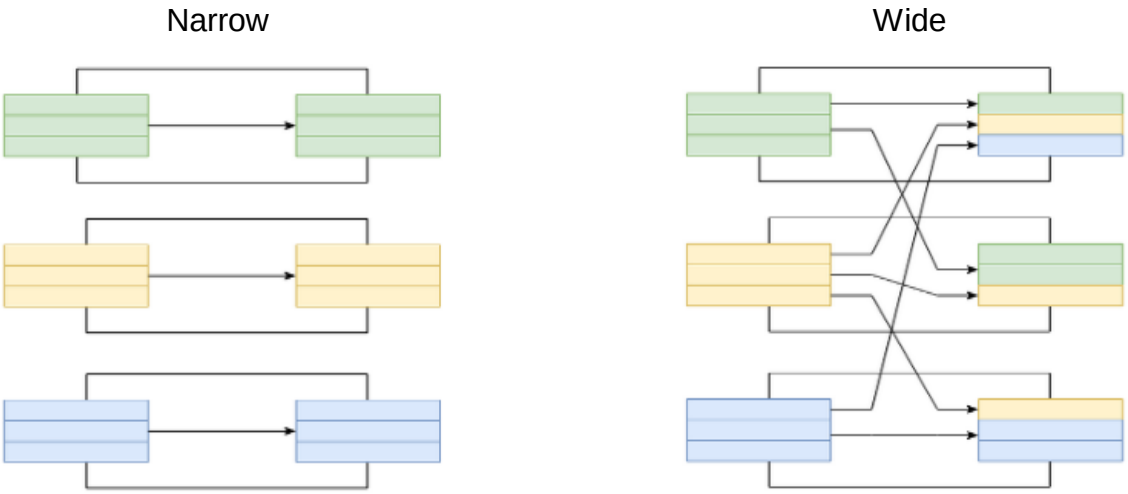


Figura 29. Tipos de transformaciones Narrow y Wide.

Tomado de (Apache Spark, 2018)

3.2.5 Componentes de Spark

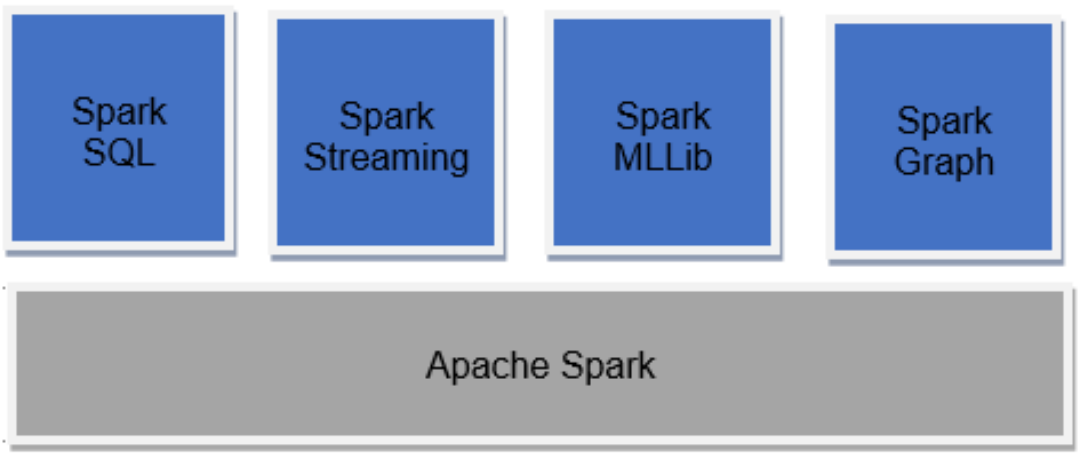


Figura 30. Estructura componentes principales de Spark.

Tomado de (OpenWebinars, 2018)

3.2.5.1 Spark Core

Base o el conjunto de librerías donde se apoyan el resto de módulos, los otros cuatro módulos. En otras palabras, se puede describir como el núcleo, la base del framework.

3.2.5.2 Spark SQL

Módulo para el procesamiento de datos estructurados y semiestructurados, con este módulo que logra que los RDD puedan ser transformados y realizar operaciones sobre ellos. Este módulo es exclusivamente para el tratamiento de los datos.

3.2.5.3 Spark Streaming

Este módulo va a permitir la ingesta de datos en tiempo real por qué se va a tener una fuente y la función del módulo es recibir los datos que emite la fuente para enviarlos a un destino.

3.2.5.4 Spark MLLib

Es una librería de Machine Learning bastante completa proporcionada por Apache Spark que contiene numerosos algoritmos que permite de una manera fácil y amigable utilizar algoritmos de Machine Learning.

3.2.5.5 Spark Graph

Es un módulo que permite el procesamiento de grafos. Se pueden crear operaciones con grafos con DAG.

3.2.6 Velocidad de procesamiento

Ejecuta cargas de trabajo 100 veces más rápido. Apache Spark logra un alto rendimiento para los conjuntos de datos y de transmisión de datos, utilizando un programador DAG de última generación, un optimizador de consultas y un motor de ejecución física. (Apache Spark, 2019).

El procesamiento de Spark se hace en la memoria, la RAM de forma natural va es más rápida que un disco duro regular. Gracias a poder hacer las operaciones directamente en la memoria, tener una buena cantidad de RAM. Spark tiene la capacidad de ir persistiendo lo que ya se va ocupando.

Spark está escrito en Scala y usa la máquina virtual de Java.

Es un framework de procesamiento unificado, esto significa que es posible hacer todas las partes del procesamiento directamente desde Spark y no se necesitan otras herramientas adicionales como era el caso de Hadoop que tiene un ecosistema muy grande.

Spark puede funcionar perfectamente encima de Hadoop o por separado.

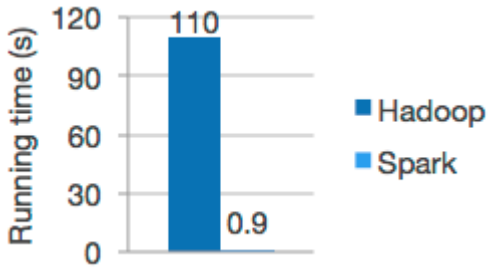


Figura 31. Regresión logística en Hadoop y Spark
Tomado de (Apche Software Foundation, 2019)

3.2.7 Funcionamiento en cualquier parte

Spark se ejecuta en Hadoop, Apache Mesos, Kubernetes, independiente o en la nube. Puede acceder a diversas fuentes de datos. Puede ejecutar Spark usando su modo de clúster independiente, en EC2, en Hadoop YARN, en

Mesos o en Kubernetes. Acceder a los datos en HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive y cientos de otras fuentes de datos. (Apache Software Foundation, 2019).Una ventaja de Spark es que tiene compatibilidad prácticamente con todas las fuentes. Si ya se posee un clúster hecho en Hadoop, únicamente se incluye Spark dentro de Hadoop, es compatible con todas las fuentes. En la actualidad Spark se ha convertido en la herramienta preponderante especialmente por esa característica.

3.3 Arquitectura Spark

En la figura anterior se puede observar que los nodos trabajadores están formados por más componentes y eso representa una aplicación y cada una de ellas posee sus propios ejecutores que ejecutan tareas en varios subprocesos, por consecuencia de ello los datos no se pueden compartir entre aplicaciones.

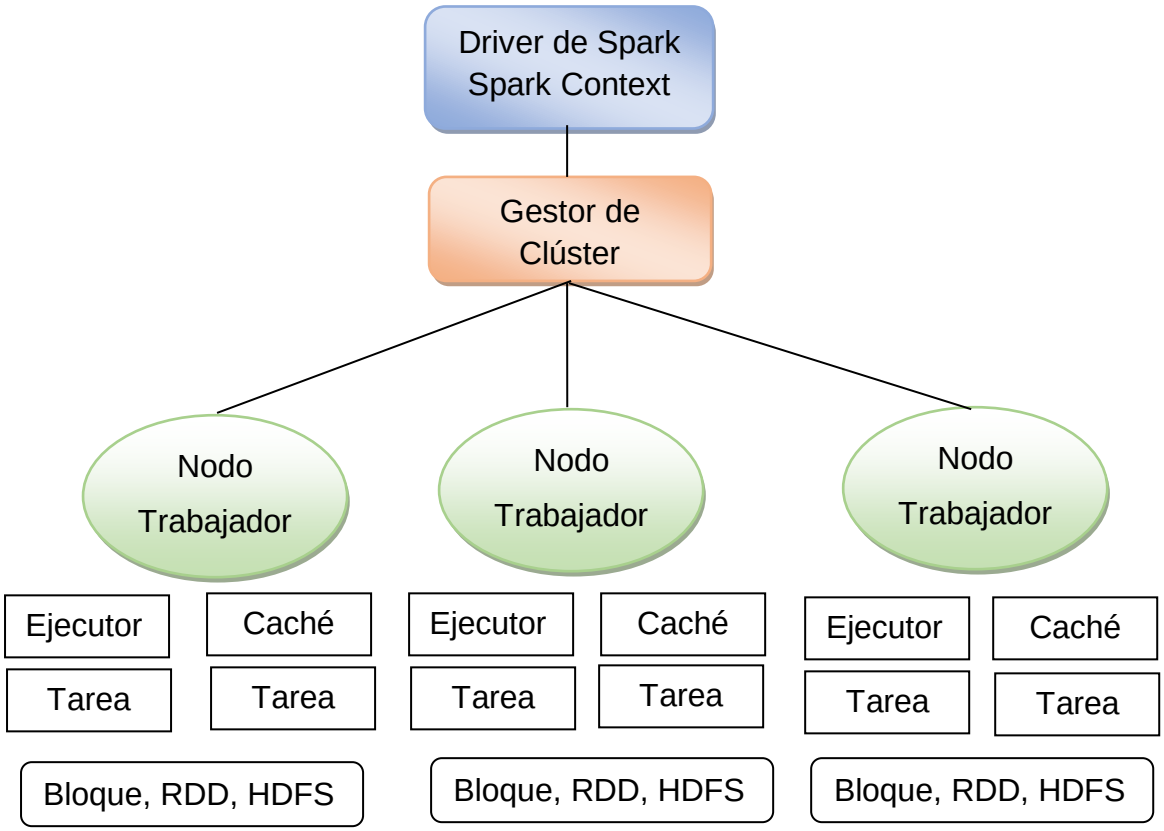


Figura 32. Arquitectura de Apache Spark.
Adaptado de: Diego Calvo.

3.3.1 API de Spark

Spark Context

Es el cerebro de la API de Apache Spark donde se realizan todas las operaciones.

Resilient Distributed Dataset (RDD)

Son datos que se generan después de intervenir con los originales su principal característica es que se dividen para mejorar su procesamiento y trabajar de forma paralela.

Existen algunas maneras de generar RDDs:

- Obtener datos de un fichero
- Obtener datos almacenados en memoria
- Obtener datos de otro RDD

4. Análisis de los resultados

En este análisis se ratificó que actualmente el uso de la herramienta de Big Data en instituciones u organizaciones de cualquier nivel ha ido incrementando. Es primordial reconocer el crecimiento masivo, la variedad y la velocidad que ahora toma la información, lo que hace imprescindible capturarla, almacenarla y analizarla para comprender la información. Sin lugar a dudas Apache ha sido la organización que más aportes realizó con la creación de plataformas que ayudan implementar Big Data. El framework Hadoop es el más utilizado para procesar grandes volúmenes de datos por medio de sus componentes HDFS y MapReduce. Otro framework de gran importancia es Spark, puede trabajar en conjunto con Hadoop. La principal característica es su velocidad ya que trabaja utilizando la memoria y no el disco.

Tabla 11.

Características MapReduce y Spark

Características	MapReduce	Spark
Desarrollado por	Google	Universidad de Berkeley
Finalidad	Procesar segmento de datos	Procesar datos en tiempo real
Lenguaje de programación	Java	Scala, Python, Java
Soporte de proceso en memoria	Si	No
Almacenamiento	Disco duro	Memoria
Tolerancia a fallos	Por tener replica de datos	Registro de transformación
Cuello de botella	Frecuente acceso a disco	Gran consumo de memoria
Capacidad de datos	102.5 Tb	100 Tb
Tiempo de proceso	72 min	23 min
Cantidad de nodos	2100	206

4.1 Análisis de las plataformas Hadoop y Spark

En este punto se realiza un análisis comparativo entre las plataformas propuestas, teniendo en cuenta que ambas realizan funciones similares y en ciertos casos resulta difícil realizar la comparación, ya que en ciertos casos procesan los datos en forma diferente.

4.1.1 Arquitectura

Spark. - El procesamiento de los datos se realiza en la memoria y se almacena en la misma. Lee desde un archivo HDFS mediante el SparkContext, después crea una estructura llamada RDD que representa datos operándose en paralelo. Según la creación de los RDD también se crea un DAG para visualizar el orden de las operaciones y sus respectivas relaciones. Aparecen los DataFrames como una interfaz complementaria para los RDD para facilitar su uso.

Hadoop. - Los archivos son divididos en bloques y según su tamaño de bloque y su factor de replicación cada bloque se repite un número de veces en el clúster. Esta información pasa al NameNode para que asigne los archivos a una serie de nodos. MapReduce está formado por una JobTracker, una vez que una aplicación está escrita en algún idioma, Hadoop acepta el JobTracker y le asigna el trabajo. Yarn es el encargado de asignar recursos al JobTracker para que este implemente, monitoree y mueva los procesos para una mayor eficiencia. Todos los resultados de los procesos se escriben en el disco HDFS.

4.1.2 Posición en la industria

Según la investigación realizada sobre las plataformas en el mundo educativo y empresarial, se pudo demostrar la supremacía de Spark con respecto a Hadoop en el procesamiento de datos. Anteriormente la técnica más utilizada era MapReduce de Hadoop teniendo el dominio del procesamiento Big Data. Actualmente tras la aparición de Spark se presenta un cambio de tendencia inclinándose hacia Apache Spark. IBM lo definió como el proyecto de open source más importante de la década (IBM, 2017).
En la figura 33 se puede demostrar que existe un mayor número de desarrolladores implicados en el proyecto de Spark.

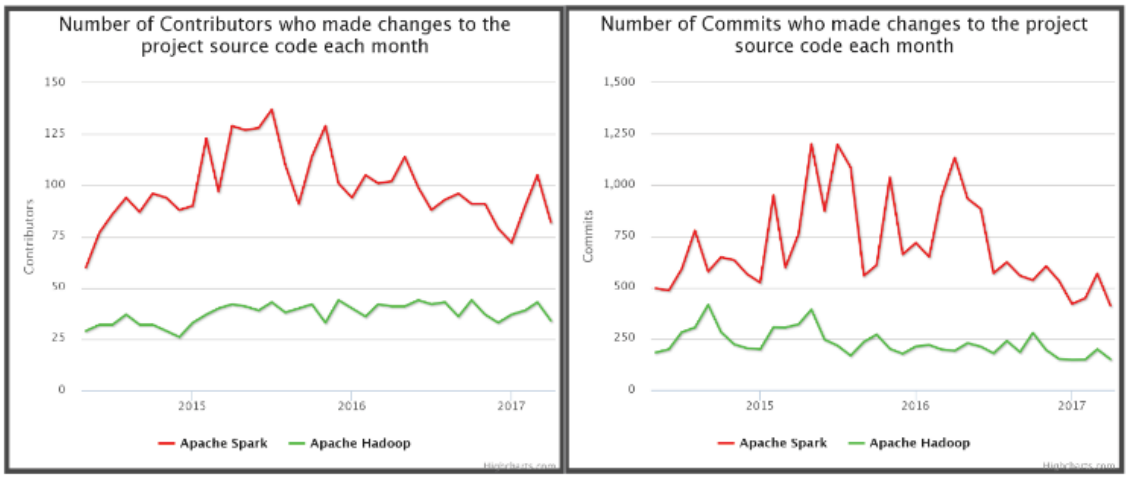


Figura 33. Desarrolladores de Spark y Hadoop.

Adaptado de (Universidad Politécnica de Madrir, 2016)

En la actualidad existe un gran numero de empresas y universidades que tienen como plataforma determinada para el procesamineto de datos a Spark. Se mencionaran algunas de las más conocidas como: Universidad de Berkeley, Amazon, eBay, IBM, Universidad de Estambul Universidad de Missorui, Samsung. En la figura 33 y la figura 34 se puede observar los gráficos que representan el desarrollo con respecto a OpenHub y GitHub.

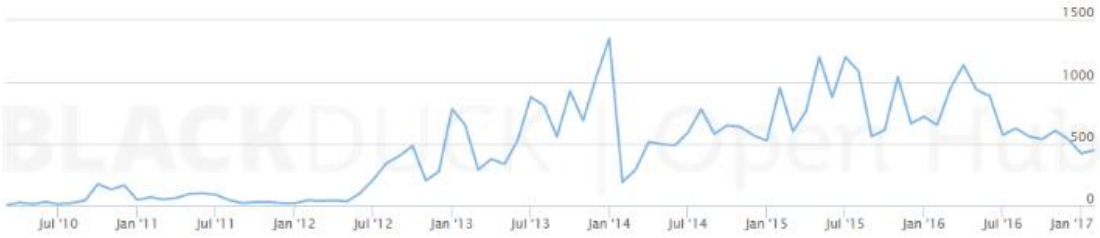


Figura 34. Histórico de aportaciones en OpenHub.

Tomado de (Apache Scala, 2018)

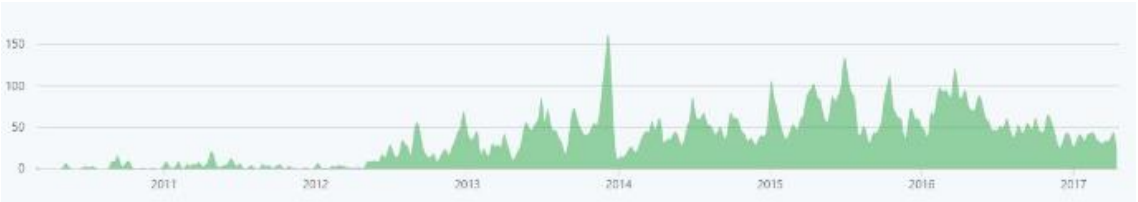


Figura 35. Histórico de aportaciones en GitHbub.

Tomado de (Apache Spark, 2018)

El lenguaje predominante en el que se escribe Spark es Scala, a pesar de estar implementado en 14 lenguajes de programación, la proporción de Spark es la siguiente:

- Por cada tres líneas de código = una línea de comentario
- Totalidad de líneas Spark 1828770
- Totalidad de líneas Hadoop 3045800

Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
Scala	794,905	301,111	27.5%	155,378	1,251,394	68.4%
Java	202,730	51,304	20.2%	39,695	293,729	16.1%
Python	67,323	51,717	43.4%	23,175	142,215	7.8%
R	27,726	27,144	49.5%	6,412	61,282	3.4%
SQL	17,232	1,356	7.3%	1,272	19,860	1.1%
CSS	15,546	508	3.2%	2,346	18,400	1.0%
XML	13,101	2,469	15.9%	768	16,338	0.9%
JavaScript	9,254	2,532	21.5%	2,040	13,826	0.8%
shell script	4,188	3,378	44.6%	1,342	8,908	0.5%
HTML	984	86	8.0%	92	1,162	0.1%
DOS batch script	442	44	9.1%	78	564	0.0%
Ruby	352	116	24.8%	114	582	0.0%
Make	302	44	12.7%	66	412	0.0%
C	38	40	51.3%	20	98	0.0%
Totals	1,154,123	441,849		232,798	1,828,770	

Figura 36. Lenguajes de programación que usan Spark.

Tomado de (Apache Scala, 2018)

4.1.2 Velocidad del proceso

Spark.- En el procesamineto de los datos, el proceso que realiza Spark tiene una gran ventaja con respecto a Hadoop. A razón de 10 a 100 veces muho más rápido que MapReduce de Hadoop. El concepto de Conjuntos Distribuidos Resilinetes, RDD que maneja Spark que permiten almacenar los datos en memoria y reutilizar los datos eficientemente ya que tiene el disco, que lo usará solo cuando sea necesario. Por lo tanto cabe reclarcar que en este aspecto, Spark supera a Hadoop gracias al proceso que se explicó anteriormente.

Databricks, la empresa creada por los fundadores de Spark tiene un record en velocidad de procesamineto obtenida en un concurso. El concurso de Sort Benchmark realizado en el 2014, Apche Spark participó en la categoría Daytona 100TB, obeniendo el primer lugar. El concurso se basaba en qué tan rápido procesa una gran volumen de registros, clasificando 100 TB usando 206 máquinas en 23 minutos. El record mundila pertenecia a un cluster de Hadoop MapReduce emplenado un tiempo de 72 minutos. Vale decir que se clasificaron la misma cantidad de datos, pero Spark lo hizo tres veces más rápido.

Tabla 12.

Comparación de la velocidad de procesamiento

	Spark	Hadoop
Tamaño de datos	100 TB	102.5 TB
Tiempo	23 min	72 min
Nodos	206	2100
Carcterísticas hardware de los nodos	32 vCores - 2.5Ghz, Intel Xeon E5-2670, 244GB memoria.	22.3Ghz, Hexcore Xeon E5-2630, 64 GB memoria.
Procesadores	6592 virtuales	504000 virtuales
Rendimiento del disco	618 Gb/s	3150 Gb/s
Red	Virtual 10 Gbps	Centro datos 10 Gbps
Velocidad de procesamiento medio	4.27 Tb/min	1.42 Tb/min
Velocidad de procesamiento por nodo	20.7 Gb/s	0.67 Gb/s

4.1.2 Rendimiento

Existe una dificultad al realizar cualquier comparativa entre Hadoop y Spark por el hecho de que los dos procesan sus datos en forma diferente. Como ya se mencionó, el rendimiento de Apache Hadoop se lo realiza en disco y el de Apache Spark lo hace en memoria. Para ello a continuación se presentan sus principales características.

Spark. - Una característica básica es que trabaja en memoria, con lo cual se consigue mucha velocidad de procesamiento, aunque también permite trabajar en disco. De esa manera si se tiene una cantidad de información muy grande y no cabe completamente en memoria. La herramienta permite almacenar parte en disco, teniendo en cuenta de no afectar la velocidad y los costos de almacenar en memoria ya que resulta más costosa que el disco. Su rendimiento puede verse mermado por la necesidad de usar aplicaciones con gran peso. Pero necesita más memoria para el almacenamiento.

El planificador de grafos DAG Scheduler es el que optimiza las etapas de ejecución. Los datos son derramados en el disco si no caben en la memoria, esto permite el buen funcionamiento con datos de cualquier tamaño. Del mismo modo, los conjuntos de datos almacenados en cache que no entran en la memoria se van al disco o se vuelven a calcular cuando sea necesario, según lo determine el nivel de almacenamiento del RDD. No es necesario Hadoop para ejecutar Spark, pero si se ejecuta en un clúster, si necesita algún tipo de sistema de archivos compartidos como el NFS.

Hadoop. - Al trabajar en disco la velocidad de procesamiento resulta un poco más lenta. Cuando los datos no son necesarios, estos se eliminan para no tener problemas en el rendimiento significativo de aplicaciones pesadas. En comparación a Spark, tiene la ventaja en requerir necesidades de almacenamiento menores. La clave de su rendimiento está en distribuir el almacenamiento y el procesamiento de la información trabajando de manera coordinada todos los equipos. Gracias a que Hadoop utiliza un clúster con millones de nodos puede brindar una gran potencia de procesamiento y tener la capacidad de almacenamiento masiva de datos.

4.1.3 Usabilidad

Spark. – Si se desea obtener esta plataforma se puede descargar a través de su página web (última versión 2.4.2). Esta es la opción más amigable con el

usuario porque proporciona API sencillas tanto para Java, Scala, Python, R. A parte de que posee un modo interactivo con el fin de poder interactuar de manera directa usuarios y desarrolladores. Se puede realizar retroalimentación en formato REPL sobre los comandos.

Hadoop. - Por otro lado, MapReduce de Hadoop puede tener Pig y Hive que funcionan como un complemento para que sea más sencillo usar para procesamiento y acceso a los datos. En este caso va a necesitar de más programación en lógicas simples. Ayuda a los administradores a especificar recursos en una cola, esto brinda un mejor control para que se configure la cantidad requerida de recursos.

Las tendencias de búsqueda se pueden ver reflejadas en la problemática para determinar que el trabajo con Big Data es mejor seguir usando Hadoop o Spark. Al explorar en Google Trends se puede visualizar las últimas tendencias en la red. Se exploran los resultados por región, al comparar los términos de búsqueda que se muestran en un mapa del mundo pintado con diferentes colores, en un periodo de tiempo seleccionado.

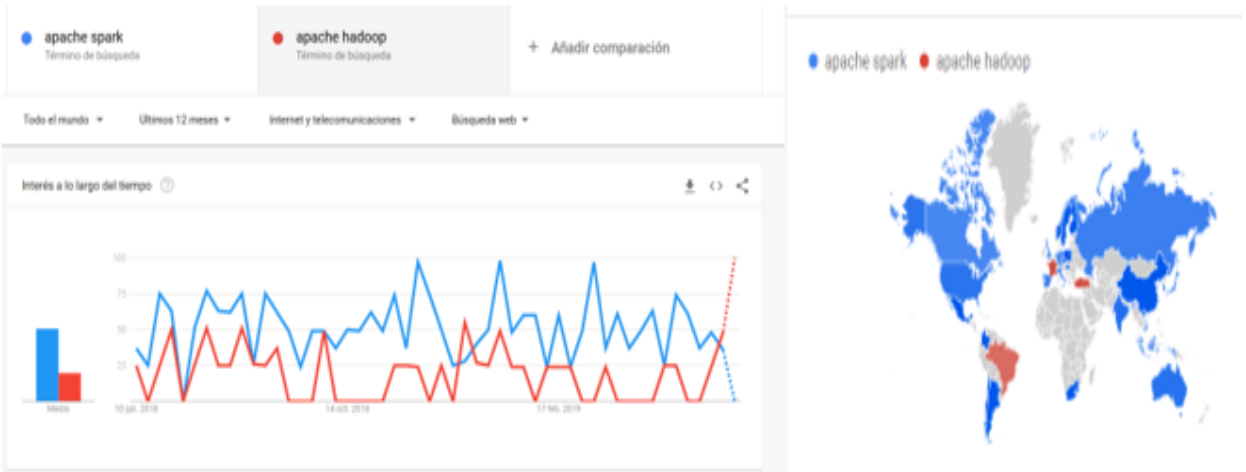


Figura 37. Impacto e interés Spark vs Hadoop.

Adaptado de (Google Trends, 2019)

4.1.4 Costos

El modelo de gestión de costos del PMI tiene ciertos componentes para realizar la planificación de los costos de un proyecto, se deben seguir los siguientes pasos para realizar un proyecto basado en Big Data.

- Planificación de la gestión de costos según el PMI
- Estimar los costos del proyecto según el PMI
- Determinar el presupuesto según el PMI
- Controlar los costos según el PMI

Actualmente la computacion en la nube es una tecnologia que brinda muchos servicios a compania, universidades, etc. Gracias a estos servicios se úeden realizar procesamiento de datos y operaciones que realizan las plataformas de Big Data.

Los dos productos son de software libre y código abierto y los dos fueron diseñados para trabajar con hardware de bajo costo, pero el análisis será planteado bajo las necesidades de hardware que tiene cada uno. Implementar un ecosistema de computación distribuida con su presencia en la nube, requiere de componentes técnicos y una capacidad de infraestructura elevado.

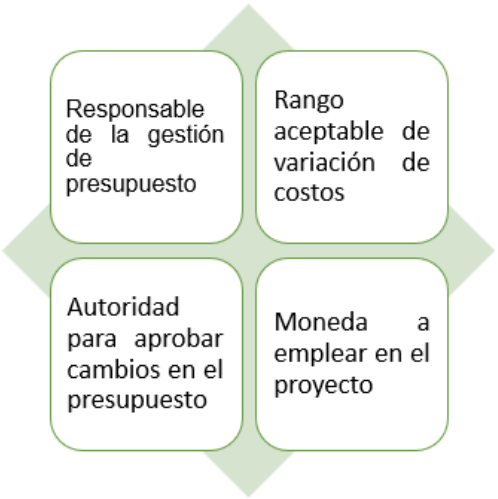


Figura 38. Componentes de un modelo de gestión de costos.
Adaptado de (PMBOOK, 2017)

Los proveedores de estos recursos como Apache han tenido un gran crecimiento mediante el uso de tecnología de bases de datos no relacionales, también está el análisis en tiempo real que se implementan con sistemas como el Internet de las cosas.

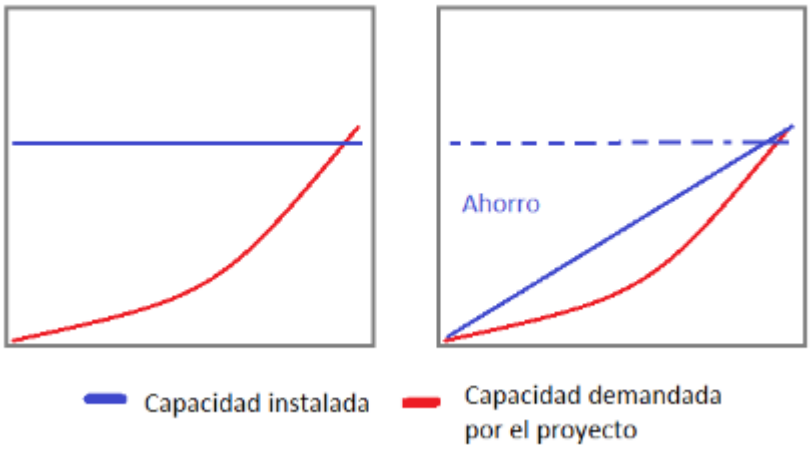


Figura 39. Comparación de capacidad instalada de servicios tradicionales con servicios modernos.

Tomado de (Universidad Militar Nueva Granada, 2016)

Spark. - Necesita más memoria RAM ya que requiere procesar una gran cantidad de datos para un rendimiento óptimo. Spark tiene su procesamiento en memoria por lo que necesita grandes cantidades de memoria para ser ejecutado. En este caso Hadoop sería la opción más económica.

Hadoop. - Se puede implementar sobre un commodity hardware normal, esta posibilidad brinda una gran flexibilidad y un ahorro muy importante. Esta es una de las mayores ventajas que ofrece Hadoop, no hace falta comprar un hardware especial ni costoso sistema de redundancia por hardware. MapReduce utiliza procesamiento basado en disco por lo que tendrá que adquirir discos más rápidos, pero de igual manera resulta más económico que los costos de Spark y su procesamiento en memoria.

Tabla 13.

Formato del control de costos.

	Entregable	C/T	C/P	EN	FE	MA	A	MA	JU	AG	DI
1	Servicios en la nube	A	Contrato Recepción Pago	0%	0%	100 %	0 %	50%		50 %	
2	Sistemas del proyecto	B	Contrato Recepción Pago	0%	100 %			100 %	100 %		
3	Acople servicios web con sistemas propios	C	Contrato Recepción Pago	10 0%		100 %					100 %
4	Infraestructura equipos de computo	D	Contrato Recepción Pago			100 %		100 %			

4.1.6 Procesamiento de datos

Spark. - Permite el procesamiento en tiempo real con un módulo de Spark Streaming. También posee su propia biblioteca de computación grafica GraphX. Spark ganó el concurso CloudSort Benchmark como el motor más eficiente en el año 2016. Consistía en clasificar 100 TB de datos con un presupuesto de $144 en recursos de la nube pública, de esa manera logró superar el anterior récord que poseía la Universidad de California, San Diego, con un costo de $451.

Posee una herramienta bastante sólida llamada Spark Streaming se la usa cuando sea necesario el procesamiento de datos en tiempo real. Puede recibir datos de diferentes fuentes como Kalka, flume, twitter. La siguiente figura detalla el proceso que se realiza el recibiendo un stream de datos que son divididos en batches para luego ser procesados por Spark y finalmente obtener batches de datos procesados.

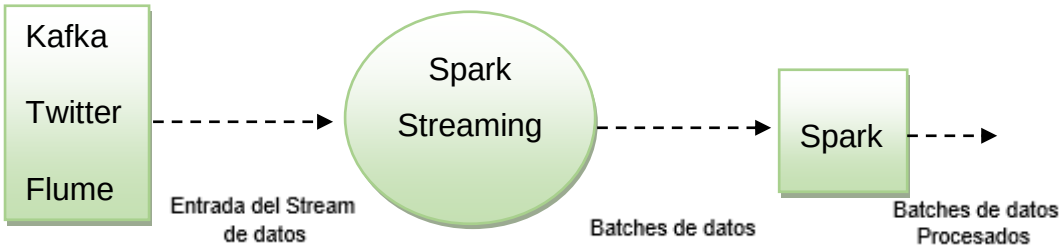


Figura 40. Proceso Spark Streaming.

Adaptado de (Escuela Técnica Superior de Ingenieros de Telecomunicación, 2016)

Hadoop. - Divide el procesamiento entre distintas máquinas para procesar enormes volúmenes de datos en tiempos eficientes y brindar un acceso rápido a los mismos. MapReduce es un motor de procesamiento por lotes siguiendo un proceso secuencial. Por un lado, Hadoop ofrece una gran fiabilidad, distribuyendo los datos y tareas entre diferentes nodos. En caso de fallar un nodo, pues la tarea se reasigna a otro. Hadoop ofrece una gran capacidad de escalado horizontal, el mismo programa se puede probar en una máquina y después en todas las que sean necesarias. Al dividir el procesamiento en varias máquinas se consigue procesar grandes volúmenes de datos en tiempos eficientes.

El punto clave de lograr un alto rendimiento es que los datos que son procesados por Hadoop se los realiza de forma secuencial para disminuir la carga de datos que se encuentran a la entrada y salida. El almacenamiento permite que los datos almacenados fuera del HDFS se asignen y dirijan desde el HDFS. Hadoop Streaming utiliza flujos estándar de Unix como la interfaz de Hadoop y su programa, por lo que puede usar cualquier idioma que pueda leer entradas estándar y escribir en la salida estándar para escribir su programa MapReduce.

4.1.7 Tolerancia a fallos

Spark. - Mientras que utilizando Spark y se llega bloquear la ejecución este tiene que procesar desde el principio. Spark es un modelo de ejecución de micro lotes, esto no tiene mucho impacto en las aplicaciones ya que los lotes pueden ser muy cortos como de 0.5 segundos por lo que no afecta la latencia. En el caso de que exista la caída de algún nodo, existe un plan de contingencia formados por mecanismos como la replicación de datos en distintos nodos y mediante el histórico de operaciones poder realizar la construcción de una línea. Esto intenta resolver la problemática de cuando una parte de la información se dañe no sea necesaria la intervención del programador, sino automáticamente las operaciones continuarán desde la última realizada antes del fallo. Logrando como finalidad que la única solución no sea mediante el control del fallo sea manual, pero al hacer uso de este mecanismo sostiene un costo en el valor computacional. Apache Spark utiliza RDD que le permite poder recuperarse ante cualquier fallo que pueda surgir en un nodo por método de recalculo del DAG, esto permite tener más tolerancia a fallos. También puede soportar un método de recuperación llamado checkpoints, parecido al que posee Hadoop.

Hadoop. - genera una gran fiabilidad por su diseño, distribuyendo los datos y tareas entre diferentes nodos. En caso de fallar un nodo, pues la tarea se reasigna a otro. Como MapReduce procesa los datos en disco, si se llegara a bloquear un proceso en ejecución no existe mucho problema ya que continua el proceso donde se dejó. La ampliación que puede realizar le permite aumentar servidores al su clúster. Uno de los factores que intervienen cuando existe un volumen muy alto de procesamiento de datos es el ancho de banda, es decir la velocidad consumida cuando se transfieren los datos. Por eso se tienen medidas como por ejemplo el HDFS que representa a su red como un árbol con sus respectivos niveles. Se definen las distancias que tendrá cada bloque mediante el uso de la jerarquía de árbol.

4.1.8 Seguridad

Spark. - necesita de Hadoop para obtener beneficios de seguridad con la finalidad de poder ejecutarse en HDFS. Se puede decir que Hadoop brinda mucha más seguridad que Spark. También se puede ejecutar en Yarn teniendo la capacidad de usar la autenticación Kerberos.

Hadoop. - Tiene el servicio Level Authorization para que los clientes obtengan permisos adecuados. Hadoop soporta autenticación Kerberos y posee Yarn. La implementación de seguridad en el entorno de Hadoop puede ser el cifrado de nivel de almacenamiento, el cifrado de campo tradicional y el enmascaramiento de datos, pero cada método tiene ciertas limitaciones. Por ejemplo, en el cifrado de nivel de almacenamiento, donde se almacena la gran cantidad de datos se cifra según el volumen del disco mientras esta en reposo en el lugar donde se almacenan los datos. Este método es un control útil en el clúster de Hadoop en el caso de que personal no autorizado pueda haber obtenido el disco en forma física. Otra técnica es el enmascaramiento de datos que es útil para confundir datos confidenciales usados con mayor frecuencia.

Estos datos enmascarados son irreversibles lo que limita su valor para muchas aplicaciones analíticas y requisitos de procesamiento. Si bien está claro que estas tecnologías proporcionan niveles de protección de los datos en Hadoop, ninguna proporciona realmente una solución de protección de extremo a extremo. La mejor solución para la seguridad de Hadoop seria aumentar los controles de infraestructura con la protección de los datos en sí. Para conseguir esto sería necesario que se anule la identificación de los datos lo más cerca posible de la fuente. Transformando los elementos de datos confidenciales con valores utilizables, pero no identificables que conserven sus mismos atributos. Hadoop deberá evitar que la información sensible llegue al HDFS de una manera vulnerable. Para implementar la seguridad basado en los datos es necesario instalar componentes de infraestructura de SecureData de Voltage.

4.1.9 Escalabilidad

Spark. - Es una plataforma escalable ya que permite la agregación medios computacionales sin que sea necesario el tener que modificar el código de las aplicaciones. Gracias a esto es posible que, en su ejecución, permanezca imperceptible al desarrollador. Spark es ejecutado en miles de nodos, en término de tamaño de datos se conoce que el grupo más grande es de 8000 de ellos. Está demostrado que hablando de tamaño de datos Spark funciona de una manera eficiente hasta petabytes. Varias cargas de trabajo de producción utilizan Spark para hacer ETL y el análisis de datos en los PB de datos.

Hadoop. - Una gran ventaja de Hadoop es la escalabilidad, especialmente cuando se despliega en plataformas de nube pública como Microsoft Azure, Amazon AWS o Google Compute Cloud. Todas las interfaces de Hadoop se clasifican según recepción y la estabilidad para mantener la compatibilidad con versiones anteriores.

Tabla 14.

Comparación entre Hadoop y Spark

	HADOOP	SPARK
Categoría	Motor básico de procesamiento de datos.	Motor de analítica de datos.
Uso	Procesamiento por lotes de un gran volumen de datos.	Procesamiento de datos en tiempo real.
Latencia	Alta latencia.	Baja latencia.
Datos	Procesamiento datos en modo batch.	Procesamiento interactivo.
Facilidad de uso	Alta seguridad	Baja seguridad en comparación a Hadoop.
Costo	MapReduce proporciona una estrategia más económica.	Más costoso que Hadoop por tener una solución en memoria.

4.2 Diferencias entre Hadoop y Spark

El modelo MapReduce lee y escribe desde un disco disminuyendo la velocidad de procesamiento, mientras que Spark reduce la cantidad de ciclos de lectura y escritura en el disco almacena sus datos en memoria por lo que su velocidad de procesamiento es más rápida.

Hadoop fue diseñado para manejar el procesamiento por lotes de manera eficiente, Spark fue diseñado datos de manera eficiente, pero en tiempo real.

Hadoop trabaja bajo un marco de lata latencia sin algún modo interactivo, Spark es un modo de computación de baja latencia con un procesamiento interactivo.

Con MapReduce el desarrollador puede procesar los datos por lotes, con Spark Streaming se puede procesar en tiempo real.

Hadoop es un sistema altamente tolerante a las fallas, mientras que Spark permite la recuperación de particiones en nodos fallidos.

Hadoop es una opción más económica por su procesamiento en disco, Spark requiere de una gran cantidad de memoria para su procesamiento por lo tanto es más costoso.

5. Conclusiones y Recomendaciones

5.1 Conclusiones

Este análisis es una guía detallada de las características de plataformas de Big Data, para la implementación de un ecosistema, aplicado a un campus inteligente. Considerando principios esenciales de la tecnología IoT, minería de datos, tomando en cuenta las necesidades reales que tiene un campus inteligente basándose en el estado actual enfocado al desarrollo de la sustentabilidad.

La siguiente conclusión que se puede extraer del presente análisis basada en la organización de un campus inteligente tras la selección de la mejor plataforma se pueda garantizar el funcionamiento eficiente, junto con un monitoreo confiable del sistema. Al mismo tiempo se trata de explotar las ventajas únicas que ofrece cada tecnología planteada en el estudio.

Este documento es una propuesta para el despliegue de un campus inteligente asociado a la innovación de sus actividades educativas con el procesamiento de sus datos recopilados para la mejora en la toma de decisiones y gestión de la universidad debido a un mayor conocimiento de toda la información obtenida. Enfocado a un punto de vista técnico para que sus posteriores estudios adicionales sean basados en este documento.

Los datos que sean recopilados por los sensores, redes y demás servicios tienen una estructura y origen diferente, se cree que el uso de Big Data es de mucha utilidad para una universidad. Creando así un valor diferenciador sobre sus competidores. Mediante el uso de esta tecnología se pueda ofrecer a sus alumnos y personal una experiencia autónoma y personalizada.

Tras la experiencia obtenida por medio del desarrollo de este análisis se describe a Spark como una plataforma mucho más intuitiva que presenta

mayor facilidad de uso, siendo una herramienta muy madura y con varias funcionalidades requeridas por diferentes empresas formándose, así como una plataforma de gran impacto en el futuro.

Se puede decir que Spark llama un poco más la atención por sus detalles con respecto a Hadoop, pero no son dos tecnologías incompatibles. Existen funcionalidades que al integrarse las dos plataformas pueden realizar funciones de acuerdo al trabajo que vayan a desempeñar, es decir las dos juntas pueden funcionar como solución.

La decisión de elegir entre Hadoop y Spark depende de los requisitos. Apache Spark es un motor de computación en clúster mucho más avanzado que el MapReduce de Hadoop, ya que puede manejar cualquier tipo de requisito, es decir, por lotes, interactivo, iterativo, streaming, etc. mientras que Hadoop solo se limita al procesamiento por lotes. Al mismo tiempo, Spark es más costoso que Hadoop con su función en memoria, que eventualmente requiere una gran cantidad de RAM. Todo depende del presupuesto y los requisitos funcionales de la universidad.

5.2 Recomendaciones

Aunque existen varios artículos o estudios que compara las características de Apache Spark y Apache Hadoop, este análisis no está encaminado hacia ese aspecto. Lo que se pretende es ofrecer un estudio de las plataformas para que pueda ser leído y analizado, después de haberlo hecho, la persona que hizo uso del miso pueda tener un mejor enfoque según las necesidades que requiere escoger el mejor y más eficiente.

Si se va a trabajar con datos estructurados la mejor solución puede ser Hadoop y no sería necesario involucrar las funcionalidades de Spark ya que aún no esta tan desarrollada en el aspecto de seguridad como lo está Hadoop.

Como ultima idea se puede decir que no siempre es importante el tamaño de los datos lo que hace esencial al Big Data. Para seleccionar la mejor plataforma se debe tomar en cuenta la manera de recolectar los datos, también que el análisis de los datos sea efectivo técnica y económicamente, eso s los que hace grande a Big Data. Se ofrecen una gran cantidad de herramientas para el procesamiento de datos para ello es necesario definir los parámetros que definen las necesidades para la adquisición de la tecnología.

Referencias

Aion N., Helmandollar L., Wang M., y Ng , J. (2012). Intelligent campus (iCampus) impact study. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, 25(7), 291-295. http://doi.10.1109/WI-IAT.2012.261*

Alvarez, M., López, G., Vázquez, E., Villagrá, V, y Berrocal, J. (2017). Smart CEI moncloa: An iot-based platform for people flow and environmental monitoring on a Smart University Campus. *Sensors (Switzerland)*, *17*(12). *http://doi.*org/10.3390/s17122856

Apache Scala. Scala como lenguaje de programación de propósito general. Recuperado el 18 de abril de https://www.openhub.net/p/scala.

Apache Spark. Motor de análisis unificado ultrarrapido. Recuperado el 18 de abril de https://github.com/apache/spark/graphs/contributors.

Buenaño, D., Villegas-CH, W., y Luján, S. (2019). The use of tools of data mining to decision making in engineering education. A systematic mapping study. *Computer Applications in Engineering Education*, *27*(3), 744–758. *http://doi.*org/10.1002/cae.22100

Camargo, J. J., Camargo, J., y Joyanes, L. (2015). Knowing the Big Data. *Revista Facultad de Ingeniería (Universidad Pedagógica y Tecnológica de Colombia)*, *24*(38), 63–77. https://doi.org/10.12053/01231129.3144

Camargo, J. J., Camargo, J. F., y Joyanes, L. (2014). Conociendo Big Data. Revista Facultad De Ingeniería, 24(38), 63. https://doi.org/10.19053/01211129.3159

Datos, M. De, Román, J. V., García, R. M. C., Jesús, J., y Rueda, G. (n.d.). (2017). Inteligencia en Redes de Comunicaciones. Mineria de Datos. Recuperado el 10 mayo de 2019 de https://www.mdpi.com/2504-3900/21/1/43

Demchenko, Y. (2013). *Defining the Big Data Architecture Framework (BDAF) Outcome of the Brainstorming Session at the University of Amsterdam*. (July). Recuperado el 20 de junio de 2019 de https://bigdatawg.nist.gov/_uploadfiles/M0055_v1_7606723276.pdf

Esumer. (2017). Big data y los nuevos manejos de la información. Recuperado el 25 de mayo de 2019 de https://www.enley.com/catalogue/big-data-y-los-nuevos-manejos-la-información/ .

Felices, R. (2017). Influencia De Las Estrategias Pasivas De La Envolvente En El Confort Térmico De Un Edificio Bioclimático. (Tesis de maestría). Universidad Politécnica de Madrid. Recuperado el 2 de junio de 2019 de http://oa.upm.es/47243/1/TFG_BORJA_RODRIGUEZ_PANOS.pdf

Hussain, S., Bang, J. H., Han, M., Ahmed, M. I., Amin, M. B., Lee, S., Parr, G. (2014). Behavior life style analysis for mobile sensory data in cloud computing through mapreduce. Sensors (Switzerland), 14(11), 22001–22020. https://doi.org/10.3390/s141122001

IBM. Explore IBM software and solutions. Analytics. Recuperado el 8 de junio de 2019 de https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html.

John Wiley y Sons Ltd. (2015). Big Data Using Smart Big Data Analytics and Metrics to Make Better Decisions and Improve Performance. Rev

Sensors. 16(6), 1-19 Recuperado el 2 de julio de 2019 de http://repository.unimilitar.edu.co/handle/10654/14920

Kumar, S. (2017) HDFS Architecture Explained Available. Recuperado el 15 de junio de 2019 de http://www.thisissanjeeva.com/tech/bigdata/2017/07/hdfs-architecture-explained

León, F., y Zapata, H. Y. (2017). *Modelo de gestión de costos en proyectos big data para Startups con enfoque PMI*. (Tesis de maestría). Universidad Militar Nueva Granada. Recuperado el 15 de junio de 2019 de https://repository.unimilitar.edu.co/bitstream/handle/10654/14411/Rodr%EDguezColmenaresJuanFernando2016.pdf;jsessionid=1CBEA95A5A4E881ABE101F72357C4F07?sequence=1

Leví, A. S., Dra, D., Maria, S., Fin, T., y Estudios, D. (2018). *Aproximación al Big Data . Análisis de su posible utilización en la universidad pública*. Universidad Politécnica de Cartagena. Recuperado el 18 de junio de 2019 de http://repositorio.upct.es/bitstream/handle/10317/7447/tfg-san-apr.pdf?sequence=1&isAllowed=y

Marques, M. B. (2014). *Análisis, uso y desarrollo experimental de herraminetas y tecnologías Open Source en Big Data*. (Tesis de maestria). Universidad Politécnica de Cataluña. Recuperado el 20 de junio de 2019 de https://upcommons.upc.edu/bitstream/handle/2117/114322/memoria.pdf?sequence=1&isAllowed=y

MicroStrategy. Enterprose Analytics and Mobility. (2019). Soluciones para la educación superior. Recuperado el 06 de marzo de 2019 de https://www.microstrategy.com/es/solutions/industries/higher-

education

Oracle. (2014). Integrated Cloud. Applications & Platform Services. Recuperado el 08 de abril de 2019 de https://oracle.com/spain/qu-es-una-base-de-datos-nosql

Pérez, C. Satín, D. (2007). Minería de Datos: Técnicas y herramientas. (1.a ed.). [version electrónica]. Recuperado el 10 de marzo de 2019 de https://books.google.com.ec/books?hl=es&lr=&id=wz-D_8uPFCEC&oi=fnd&pg=PR4&dq=miner%C3%ADa+de+datos&ots=TiY0sh8zbL&sig=TTwzNB-Oy4rcLI-kUG5yt8Rus2w&redir_esc=y#v=onepage&q=miner%C3%ADa%20de%20datos&f=false

PMBOK. (2017). Project Management Body of Knowledge 5Th -Capitulo 7. Recuperado el 21 de junio de 2019 de https://www.pmi.org/pmbok-guide-standards/foundational/pmbok

Popoola, S. I., Atayero, A. A., Badejo, J. A., John, T. M., Odukoya, J. A., y Omole, D. O. (2018). Learning analytics for smart campus: Data on academic performances of engineering undergraduates in Nigerian private university. *Data in Brief*, 76–94. https://doi.org/10.1016/j.dib.2017.12.059

Requena, A. OpenWebinars. Big Data Apache Spark. Recuperado el 20 de mayo de 2019 de: https://openwebinars.net/blog/que-es-apache-spark/

Rocha, Á., y Guarda, T. (2018). *Erratum to: Proceedings of the International Conference on Information Technology & Systems*, 1(3), E1–E1. https://doi.org/10.1007/978-3-319-73450-7_110

Sagiroglu, S. Sinanc, D. Big data: a review. In: International Conference on Collaboration Technologies and Systems (CTS), pp. 42–47 (2013).

https://doi.10.1109/ACCESS.2019.2894129

Saksena, S. (2018). How Smart Campuses transform student engagement, Health, productivity and retention. Educause. Recuperado el 18 de junio de https://events.educause.edu/annual-conference/2018/agenda/how-smart-campuses-transform-student-engagement-health-productivity--retention

Samadi, Y., Zbakh, M., & Tadonki, C. (2017). Comparative study between Hadoop and Spark based on Hibench benchmarks. Proceedings of 2016 International Conference on Cloud Computing Technologies and Applications, CloudTech 2016, 267–275. https://doi.org/10.1109/CloudTech.2016.7847709

Micro Focus. (2017). Sheet, D. (n.d.). Voltage SecureData for Hadoop and IoT. l. Recuperado el 28 de junio de 2019 de https://www.microfocus.com/media/data-sheet/voltage_securedata_for_hadoop_and_iot_ds.pdf

Silva, B. N., Khan, M., Jung, C., Seo, J., Muhammad, D., Han, J., Han, K. (2018). Urban planning and smart city decision management empowered by real-time data processing using big data analytics. Sensors (Switzerland), 18(9), 6–12. https://doi.org/10.3390/s18092994

Villegas-ch, W., Palacios-pacheco, X., y Luján-mora, S. (2018). *Un framework de big data aplicado a la mejora del aprendizaje en un smart campus*. 1–22. https://doi.10.1002/cae.22100

White, T. (2012). The Definitive Guide Hadoop: Storage and Analysis at Internet Scale. (3. ed). O Reilly. Recuperado el 19 de junio de 2019 de https://www.isical.ac.in/~acmsc/WBDA2015/slides/hg/Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition.Jan.2012.pdf

Winshuttle. (2016). Big Data y la historia del almacenamiento de la información. Recuperado el 06 de marzo de 2019 de https://www.winshuttle.es/big-data-historia-cronologica/