Universitat Oberta de Catalunya uoc.edu

ARQUITECTURA BIGDATA EN EL AMBITO DE SEGURIDAD PARA EL NOTARIADO ESPAÑOL

GIOVANNY ANDRES PALACIOS CUARTAS

MASTER UNIVERSITARIO EN SEGUIRDAD DE LAS TECNOLOGIAS DE LA INFORMACION Y DE LAS COMUNICACIONES

BIGDATA Y SEGURIDAD

Consultor: ENRIC HERNANDEZ

Profesor: VICTOR GARCIA FONT

31 DE DICIEMBRE DE 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Arquitectura Bigdata en el ámbito de la seguridad para el notariado Español*
Nombre del autor:	*GIOVANNY ANDRES PALACIOS CUARTAS*
Nombre del consultor/a:	*ENRIN HERNANDEZ*
Nombre del PRA:	*VICTOR GARCIA FONT*
Fecha de entrega (mm/aaaa):	12/2018
Titulación::	*MASTER UNIVERSITARIO EN SEGURIDAD DE LAS TECNOLOGIAS DE LA INFORMACION Y DE LAS COMUNICACIONES*
Área del Trabajo Final:	*BIGDATA Y SEGURIDAD*
Idioma del trabajo:	*ESPAÑOL*
Palabras clave	*BIGDATA, PROCESAMIENTO, HADOOP*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo del presente proyecto, consiste en proponer una alternativa de arquitectura Big Data para el análisis y monitoreo de grandes cantidades de registros en tiempo real generados por los servidores del notariado español cuya finalidad es generar un sistema de monitoreo en tiempo real que permita la detección de eventos de seguridad en sus servidores, tales como explotación de vulnerabilidades, accesos no autorizados, trafico anómalo y comportamientos inusuales.

El proyecto inicia presentando la tecnología Hadoop, su funcionamiento y finalidad, para continuar explicando un portafolio de herramientas que comprende su ecosistema, luego se analiza una tecnología que en la actualidad es de bastante uso como es Spark. El proyecto se desarrolla clasificando cada tecnología en todo el ciclo de implementación de una arquitectura Big data, en la cual, tiene como procesos relevantes los siguientes: Ingesta de datos, procesamiento, almacenamiento en clúster y visualización de resultados.

Por último, finaliza con la selección y escogencia de una posible arquitectura, la propuesta contempla la especificación de cada tecnología para los procesos relevantes de una arquitectura Big Data, esta elección de tecnologías tuvo como principal factor cumplir la finalidad planteada y las características del caso de uso (generación de datos en cantidades masivas, tratamiento y análisis en tiempo real- Stream). Como conclusión, se puede indicar que existe una gran cantidad de herramientas que se pueden utilizar para la aplicación de la solución de caso planteado y en general para Big Data, de igual forma, su uso en el aspecto de la seguridad juega un papel importante para poder analizar en tiempo real posibles eventos de

seguridad.

Abstract (in English, 250 words or less):

The objective of this project is to propose a Big Data architecture alternative for the analysis and monitoring of great volume numbers of real-time records generated by Spanish notary servers whose purpose is to generate a real-time monitoring system that allows detection of security events on their servers, such as exploitation of vulnerabilities, unauthorized access, anomalous traffic and unusual behavior.

The project begins by presenting the Hadoop technology, his operation and purpose, to continue explaining a portfolio of tools that comprise its ecosystem, then analyzing a technology that is currently widely used as Spark. The project is developed by classifying each technology throughout the implementation cycle of a Big Data architecture, in which the following processes are relevant: Data intake, processing, storage in clusters and visualization of results.

Finally, it ends with the selection and choice of a possible architecture, the proposal includes the specification of each technology for the relevant processes of a Big Data architecture, this choice of technologies had as main factor to fulfill the purpose and characteristics of the case of use (generation of data in massive quantities, treatment and analysis in real time – Stream. In conclusion, it can be indicated that there is a large number of tools that can be used for the application of the proposed case solution and in general for Big Data, likewise, its use in the security aspect plays an important role for to be able to analyze in real time possible security events.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

1. INTRODUCCION

1.1. CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

Ancert es una empresa que maneja el Soporte de tecnología del notariado español, estas notarías se encuentran distribuidas por el país, cada notaría cuenta con un servidor administrado y gestionado por Ancert (aproximadamente 3000 servidores). Cada servidor, envía una gran cantidad de información a la central de forma continua, información referente a: eventos de Seguridad (conexiones remotas, intentos de intrusión) e información relevante para su respectivo análisis y estadísticas de servidor, sin embargo, por la cantidad de información enviada y la continuidad de la misma, no es posible su procesamiento en tiempo real y se debe realizar muestreos para su análisis. Es de vital importancia para Ancert, diseñar una infraestructura optima que pueda analizar y gestionar toda la DATA enviada en tiempo real, tener la capacidad para procesar datos variados, a gran velocidad y en gran volumen que sea de bajo costo y con un pilar fundamental que es la seguridad.

1.2. OBJETIVO PRINCIPAL

- ✓ Proponer una arquitectura BIG DATA para el procesamiento de la información de eventos de seguridad para el notariado español.

1.3. OBJETIVO SECUNDARIOS

- ✓ Conocer las diferentes tecnologías Big data existentes (Hadoop, Spark)
- ✓ Conocer y evaluar el Ecosistema Hadoop como parte fundamental de la arquitectura BIG DATA.
- ✓ Seleccionar las herramientas del ecosistema que mejor se adecuen al caso de uso analizado.
- ✓ Comparar las diferentes alternativas para implementar la arquitectura propuesta.

1.4. ENFOQUE Y METODO SEGUIDO

De acuerdo a la problemática expuesta y el objetivo de la misma, se decide como metodología de investigación elegida para el desarrollo del proyecto la sintética, esto debido a que a partir de varios elementos (pool de herramientas para la arquitectura a implementar) por separado se pretende llegar a un resultado concreto, esto quiere decir, a definir una arquitectura seleccionada que mejor se adecue a las características del proyecto.

Para cumplir a cabalidad esta metodología, inicialmente se plantearán todas las tecnologías a utilizar durante el desarrollo del proyecto, realizando previamente un análisis frente a sus características, con el objetivo de determinar cuáles son las más viables para la implementación.

En la segunda fase, la cual se centra en el análisis de la arquitectura a implementar, una vez definidas las herramientas a utilizar y teniendo claro los objetivos a cumplir, se iniciará con el análisis y evaluación del ambiente Big data propuesto, siguiendo así de esta manera la finalidad de esta metodología, donde se parte de lo abstracto para llegar a lo propuesto.

Fuentes de Información. Dentro de las fuentes de información que se van a utilizar para el desarrollo del proyecto, se encuentran:

- ✓ **Fuentes primarias.** Libros especializados en el manejo del tema de Big Data, que lograrán identificar conceptos claves para conseguir una implementación óptima; también artículos de internet, los cuales se tomarán como guía para escoger e instalar las herramientas más adecuadas para la construcción del ambiente.
- ✓ **Fuentes secundarias.** La participación en conferencias sobre Big Data, tanto de manera presencial (de ser posible) como de manera virtual, adicionalmente la asesoría con personas que tengan altos conocimientos acerca del tema, logrando así un óptimo desarrollo, de igual forma, reuniones periódicas virtuales con el tutor para realizar los ajustes y despejar dudas que se encuentren.

1.5. ACTIVIDADES O TAREAS A REALIZAR PARA CUMPLIR CON LOS OBJETIVOS.

- ✓ Investigar arquitectura BIG DATA: Ventajas y Desventajas
- ✓ Conocer la infraestructura actual, deficiencias.
- ✓ Analizar el tipo de logs remitidos por los servidores.
- ✓ Conocer ecosistema Hadoop.
- ✓ Creación de logs automáticos.
- ✓ Conocer complemento SPARK.
- ✓ Conocer tecnología de lectura de Datos KAFKA.
- ✓ Definición de arquitectura a implementar.

1.6. PLANIFICACIÓN DEL TRABAJO

ACTIVIDAD	Fecha
CONTEXTUALIZACION	**19/09/2018 – 18/10/2018**
Comunicación Ancert (actividad Continua)	19/09/2018 a 04/10/2019
Planificación Inicial	02/10/2018 a 09/10/2018
Búsqueda de información sobre Big DATA	08/10/2018 a 18/10/2018
Comparación de arquitecturas Big DATA	12/10/2018 a 18/10/2018
ANALISIS HERRAMIENTAS HADOOP	**15/10/2018 – 03/11/2018**
Búsqueda y análisis Hadoop	15/10/2018 a 23/10/2018
Análisis Componentes Hadoop	21/10/2018 a 29/10/2018
Ecosistema Hadoop y sus herramientas	28/10/2018 a 02/11/2018
Búsqueda y comparativa de soluciones MapReduce - Spark	01/11/2018 a 03/11/2018

DISEÑO ARQUITECTURA PROPUESTA	**30/10/2018 – 15/12/2018**
Análisis detallado del caso de Uso	30/10/2018 a 06/11/2018
Diseño arquitectura propuesta	07/11/2018 a 14/11/2018
Selección de herramientas para cada fase de la arquitectura Big Data	15/11/2018 a 05/12/2018
Comparativa de implementación de arquitectura propuesta Hadoop	05/12/2018 a 15/12/2018
ANALISIS DE RESULTADOS	**15/12/2018 – 05/01/2019**
Análisis de Arquitectura propuesta	15/12/2018 a 18/12/2018
Conclusiones	18/12/2018 a 31/12/2018
Presentación / creación de video - de Memoria TFM	01/01/2019 a 05/01/2019

2. ESTADO DEL ARTE

El proyecto se enmarca en una aplicación de arquitectura BIG DATA, y la evaluación de un ecosistema Hadoop, para lo cual, es bueno tener presente como se encuentra esta tecnología actualmente.

2.1. BIG DATA

Actualmente, se generan muchos datos y las empresas exigen saber no sólo lo que sucede en la actualidad, sino también lo que va a pasar en un futuro, requiriendo niveles de servicio mucho más exigentes que hace unos años. Es precisamente en esta tesitura donde se encuentra Big Data hoy día.

El concepto big data hace referencia a los sistemas que manipulan grandes conjuntos de datos, también conocidos como data sets. Entre sus principales cualidades se encuentran la heterogeneidad y la volatilidad, como datos que son. Sin embargo, son su volumen y su velocidad de generación las que plantean mayores dificultades a la hora de trabajar a un nivel big data en el entorno empresarial. Los retos tienen que ver con:

- Captura de datos.
- Almacenamiento de tales volúmenes de información.
- Capacidad de realizar búsquedas eficientes.
- Compartición.
- Posibilidad de llevar a cabo análisis efectivos.
- Visualización de los datos.

En la actualidad, la cantidad de datos que se generan es abismal y de una casuística extremadamente compleja para su análisis. Como hemos comentado, las empresas cada vez exigen que el análisis sea lo más cercano posible al tiempo real. Y en Big Data está la clave, al traducirse el mismo en las variables de velocidad, variedad y volumen que requiere el mercado actualmente, igualmente, hasta hace unos años lo más importante era la transacción de información, pero, actualmente, esto está cambiando. Cada vez se da más importancia a la interacción constante con el mundo que nos rodea. Las empresas tienen una única opción: aprovecharse de toda la información que se genera.

Ilustración 1. Caracteristicas BigData

Estamos acostumbrados a saber lo que pasó, pero hoy nos interesa más conocer lo que pasará: si la empresa, con su producto o servicio, seguirá siendo el gusto de los consumidores o si se hablará de ella bien o mal en las redes sociales; aspectos todos ellos que requieren de nuevos modelos de análisis mucho más complejos que lo que se podía experimentar hasta ahora.

Esta nueva realidad ha motivado nuevos requerimientos por parte de las empresas en relación con el análisis de datos. Y, precisamente, para poder analizar toda esa información de que hoy se dispone, lo que anteriormente se conocía como Business Intelligence actualmente requiere de un nuevo modelo de análisis: Big analytics, el único que permite dar forma al Big Data.

El Big Data conlleva desafíos que pueden ser resueltos con el Ecosistema Hadoop.

2.2. HADOOP

Hadoop es un sistema de código abierto que se utiliza para almacenar, procesar y analizar grandes volúmenes de datos, aislando a los desarrolladores de todas las dificultades presentes en la programación paralela. Hadoop cuenta con todo un ecosistema de ayuda que, además de distribuir el fichero en sus nodos (que no son más que ordenadores con commodity-hardware), hace posible ejecutar procesos en paralelo; disponiendo también de módulos de control, monitoreo y consultas. Empiezan entonces aparecer distintos add- ons que ayudan a poder trabajar, manipular y monitorizar la información que se está guardada sobre Hadoop.

Los componentes básicos de Hadoop son los siguientes:

- **HDFS**
  Es el sistema de archivo distribuido que permite que el fichero de datos no se guarde en una única máquina, sino que la información se distribuya en distintas. Escrito en JAVA.
- **MAPREDUCE**
  Se trata de un framework de trabajo que permite aislar al programador de todas las tareas propias de la programación en paralelo. Es decir, hace posible que un programa escrito en los lenguajes de programación más común se pueda ejecutar en un cluster de Hadoop. La gran ventaja es el poder usar el lenguaje o las herramientas más adecuadas para la tarea concreta que se ha de realizar en cada momento.
  Principales características del MapReduce:

  • Distribución y paralelización (automáticas).
  • Tolerancia a fallos y a redundancias.
  • Transparencia: su funcionamiento interno y su mantenimiento son transparentes para los desarrolladores. Es decir, que sólo tienen que programar la lógica de negocio del algoritmo, en vez de necesitar invertir tiempo gestionando errores o parámetros de la computación distribuida.
  • Escalabilidad horizontal: permite que, si se necesita más potencia de computación, baste con añadir más nodos en el clúster.
  • Localización de los datos: se desplaza el algoritmo a los datos y no al contrario, como suele suceder en sistemas distribuidos tradicionales.
  • Dispone de herramientas de monitorización.

2.3. Utilidades de Hadoop

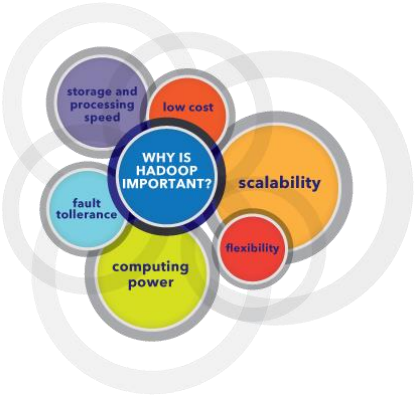
Hadoop es un sistema que se puede implementar sobre hardware a un costo relativamente bajo, siendo a su vez totalmente gratuito para software.

Ello ha comportado que, toda la información que antes las empresas no podían procesar por las limitaciones de la metodología existente, hoy pueda ser procesada gracias a Hadoop. de esta forma se puede, no sólo obtener información nueva, sino también descubrir y aplicar otro tipo de análisis como, por ejemplo, una regresión lineal, sobre millones de registros de su histórico.

Éste es el principal motivo detrás de la rápida propagación de su uso entre las empresas, que ven que, con una inversión relativamente baja, pueden afrontar nuevos retos y problemáticas que antes no podían afrontar y con un ROI muy rápido.

A su vez, para minimizar los riesgos de su aplicación, existen en el mercado distintas distribuciones de Hadoop con soporte 24/7 que ayudan a no depender de la comunidad Open Source, lo que ha contribuido a impulsar su adopción en entornos productivos

Entre sus puntos clave se encuentran su capacidad de almacenamiento y procesamiento local. Partiendo de ellos:



• Consigue escalar desde unos pocos servidores hasta miles de máquinas, todas ellas ofreciendo idéntica calidad de servicio.
• Permite el procesamiento distribuido de grandes conjuntos de datos en clusters de computadoras utilizando modelos sencillos de programación.

Se trata, en definitiva, de un proyecto de desarrollo de software orientado hacia la computación distribuida, donde la escalabilidad y la fiabilidad son los dos atributos más importantes. En otras palabras, Hadoop completa el círculo, erigiéndose en complemento perfecto de big data porque:

• Simplifica la interacción con su aportación informativa.
• Economiza los procesos.
• Palia las carencias que big data puede presentar de cara al usuario.

Conforme ha evolucionado el marco de procesamiento distribuido Hadoop, ha llegado a incluir mucho más que su núcleo original, que consistía en el sistema de archivos distribuido Hadoop (HDFS) y el entorno de programación MapReduce. Entre una serie de nuevos componentes del ecosistema Hadoop, una tecnología ha adquirido una especial atención: el motor de procesamiento de datos en memoria Spark. Spark está reemplazando a MapReduce en un número creciente de trabajos de procesamiento por lotes en los conjuntos de Hadoop; sus defensores afirman que puede ejecutarlos hasta 100 veces más rápido:

Como podemos evidenciar la tecnología a implementar conllevo un análisis exhaustivo de las posibles soluciones.

3. CONTEXTUALIZACION

3.1. BIG DATA

Dentro del proyecto planteado, la solución más idónea para para solucionar la problemática expuesta, es diseñar una arquitectura escalable de BIG DATA donde se puedan procesar en tiempo real el volumen de información enviado por los servidores respectivos y la velocidad de procesamiento sea óptima. Para este apartado, se expone como primera instancia conocer que es Big data (ver apartado Estado del arte del presente documento) y las fases para implementar arquitecturas escalables de este Tipo e implementándolas en entornos virtualizados para validación y evaluación de la arquitectura.

Volumen	Velocidad	Variedad	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	Correlaciones
Tablas y Archivos	Procesos	Probabilística	Responsabilidad	Hipótesis

Ilustración 2. Atributos de BigData

Fuente: UNIVERSITY OF AMSTERDAM. Defining the Big Data Architecture Framework [en línea]. Ámsterdam: Yuri Demchenko [citado 22 septiembre, 2013]. Disponible en internet: <http://bigdatawg.nist.gov/_uploadfiles/M0055_v1_7606723276.pdf >

La arquitectura Big Data está compuesta generalmente por cinco capas: **recolección de datos**, **almacenamiento**, **procesamiento de datos**, **visualización** y **administración**. Debido a las nuevas necesidades cada uno de estos pasos ha ido adaptándose y aportando nuevas tecnologías a la vez que abriendo nuevas oportunidades.

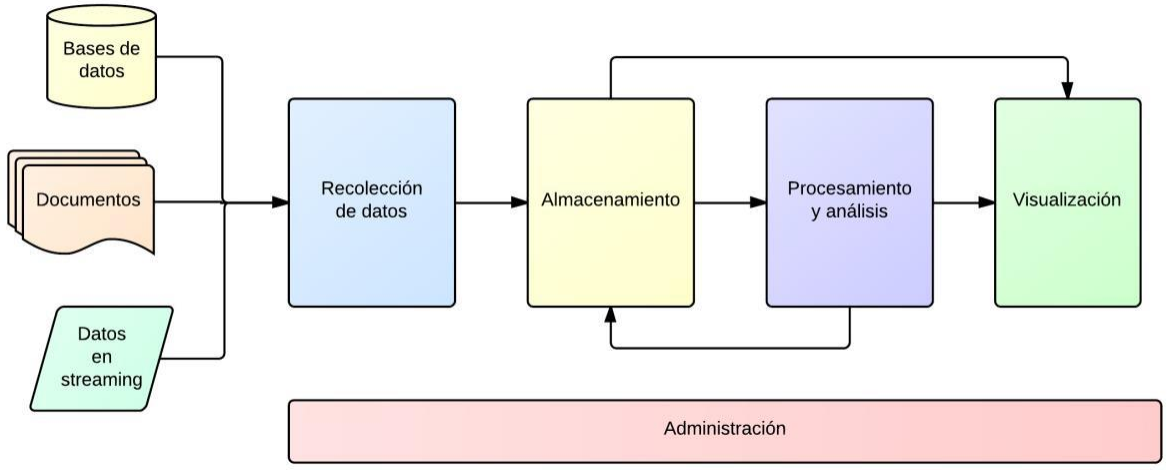


Ilustración 3. Capas BigData

3.1.1. RECOLECCIÓN DE DATOS

En esta etapa el sistema debe conectarse a las fuentes de información y extraer la misma. Las herramientas de recolección de datos pueden dividirse en dos grupos, dependiendo de cómo se conecten al origen de los datos:

1. **Batch o por lotes**: se conectan de manera periódica a la fuente de datos buscando nueva información. Generalmente se usan para conectarse a sistemas de ficheros o bases de datos, buscando cambios desde la última vez que se conectaron.

2. **Streaming o por transmisión en tiempo real**: están conectados de manera continua a la fuente de datos, descargando información cada vez que ésta transmite. Se acostumbra a usar para monitorización de sistemas -**para aumentar la seguridad y la detección de fallos**- y descargar información en tiempo real. Este es el modelo de la problemática y/o caso de uso expuesto.

3.1.2. ALMACENAMIENTO

La capa de almacenamiento tiene, dos elementos básicos: el sistema de ficheros y la base de datos. Hasta hace poco los sistemas de tratamiento de la información se centraban principalmente en las bases de datos, pero, debido a que en los sistemas Big Data se busca la mayor variedad posible -las bases de datos acostumbran a ser poco flexibles-, los sistemas de ficheros han cobrado mayor importancia.

3.1.3. PROCESAMIENTO Y ANÁLISIS

Una vez se tienen los datos almacenados, el siguiente paso en un sistema Big Data es analizar y/o procesar la información para obtener a los resultados deseados.

3.1.4. VISUALIZACIÓN

Los resultados a visualizar del procesamiento.

3.2. COMPARACIÓN DE ARQUITECTURAS BIG DATA

La aparición del concepto Big Data ha propiciado la creación de varios paradigmas de programación para el proceso de datos que intentan ofrecer un acercamiento a una solución para Big Data. Estos paradigmas han terminado caracterizando las arquitecturas Big Data, adaptando el resto de capas para funcionar de forma óptima. Los dos paradigmas que centran el desarrollo de aplicaciones son MapReduce y las llamadas Massive Parallel Procesing (o MPP), ambas con aspectos en común pero bien diferenciadas.

Para la comparativa entre los dos paradigmas se ha creado una tabla (Tabla 2) con las valoraciones de distintos aspectos que se consideran importantes en un sistema Big Data, ya que se tratan de

los puntos en los que una solución debería destacar (Las cinco V). Las valoraciones van del 1 (peor) a 3 (mejor). Al ser una comparación teórica no se pueden hacer valoraciones más allá de lo conocido. los aspectos valorados han sido los siguientes:

- ✓ **Velocidad**: los tiempos de respuesta en las operaciones, procesos y consultas a realizar. La valoración es:
  - **1**: los procesos tardan del orden de horas.
  - **2**: los procesos tardan del orden de minutos.
  - **3**: los procesos tardan del orden de segundos.
- ✓ **Volumen**: cuál es el volumen de datos con el que cada sistema puede llegar a trabajar. La valoración es:
  - 1: no permite trabajar con volúmenes de datos del orden de los gigabytes.
  - 2: no permite trabajar con volúmenes de datos del orden de los terabytes.
  - 3: no tiene problemas en trabajar con volúmenes del orden de zettabytes.
- ✓ **Escalabilidad**: la facilidad y la influencia de escalar la infraestructura. La valoración es:
  - 1: no son escalables en marcha, hay que parar el sistema y reconfigurar todo.
  - 2: son escalables sin tener que reconfigurar ni parar ningún nodo pero requiere de una especificación de hardware mínima.
  - 3: son escalables independientemente del hardware que se esté añadiendo.
- ✓ **Variedad**: con qué y con cuántos tipos de datos puede trabajar el sistema. La valoración es:
  - **1**: solo trabaja con datos estructurados.
  - **2**: admite datos semi estructurados.
  - **3**: puede trabajar con datos no estructurados.
- ✓ **Variabilidad**: cómo reacciona un sistema a un cambio en los orígenes de datos o en el significado de su información. La valoración es:
  - **1**: no permite modificar el esquema creado desde el inicio.
  - **2**: permite variar o ampliar el esquema.
  - **3**: es independiente de esquema de datos.
- ✓ **Productividad**: qué nivel de productividad puede llegar a tener un sistema teniendo en cuenta las tecnologías que usa, su implantación actual. La valoración es:
  - **1**: son tecnologías nuevas y en constante evolución, que implican un proceso de aprendizaje y una manera de trabajar distinta.
  - **2**: tecnologías nuevas, pero con una curva de aprendizaje correcta y una manera de trabajar parecida a la ya existente.
  - **3**: a pesar de ser tecnologías nuevas son totalmente compatibles con la forma de trabajar actuales -lenguajes de programación, consulta, herramientas de terceros.
- ✓ **Coste:** el precio tanto en mantenimiento como en la adquisición de la infraestructura y las licencias de software. La valoración es:
  - **1**: las soluciones basadas en este paradigma suelen ser costosas y con un mantenimiento alto.
  - **2**: sus soluciones son costosas pero tienen un mantenimiento fácil.
  - **3**: tiene soluciones flexibles, que se ajustan a los presupuestos, y no requieren demasiado mantenimiento.

Comparativa de acuerdo al anterior análisis

ITEM	MapReduce	MPP
Velocidad	2	3
Volumen	3	2
Escalabilidad	3	2
Variedad	3	1
Variabilidad	3	1
Productividad	1	3
Coste	3	1
Total	**18**	**13**

Ilustración 4. Comparativa MapReduce vs MPP

- ✓ **Velocidad:** los sistemas MPP son indiscutiblemente más rápidos -del orden de varios segundos en consultas sencillas- al tener los datos ya estructurados y preparados para la consulta mediante índices. De todas formas, las arquitecturas MapReduce ofrecen un rendimiento escalable linealmente, de manera que aumentando el número de nodos aumentaría también la velocidad, por lo que al tratar grandes volúmenes de datos podría llegar a igualar a los sistemas MPP.
- ✓ **Volumen:** las MPP están penalizadas por la creación de índices, que ocupan espacio y además limitan el crecimiento, un problema heredado de los SGBD relacionales. De todas formas al ser sistemas escalables, puede que no aumenten el rendimiento como MapReduce (que no cuenta con esta limitación) pero sí que permiten una gran cantidad de datos. Las soluciones MPP empiezan a ser sistemas limitados a partir de volúmenes de datos superiores al orden de los gigabytes y terabytes, mientras que MapReduce puede llegar a tratar volúmenes de zettabytes.
- ✓ **Escalabilidad:** los sistemas MapReduce son escalables a todos los efectos. Se puede incrementar el número de nodos incluso añadiendo nuevos con distintas especificaciones (la ejecución de cada nodo es independiente de las demás). En el caso de las MPP, se requiere unas especificaciones más exigentes y uniformes, además de que añadir un nodo implica la reorganización de los metadatos o índices.
- ✓ **Variedad:** al aceptar datos no estructurados, las arquitecturas MapReduce admiten una mayor variedad de formatos para los datos. Las MPP en cambio requieren de datos estructurados.
- ✓ **Variabilidad:** tal y como pasa con la variedad, MapReduce está preparado para aceptar cambios de cualquier tipo, mientras que las MPP están ligadas al modelo de datos que se diseña al crear la base de datos, dificultando la modificación a posteriori.
- ✓ **Productividad:** las MPP son bases de datos relacionales que llevan usándose durante décadas, por lo que la productividad es mayor al no tener el usuario que aprender nuevos lenguajes (además de ser un lenguaje muy sencillo). MapReduce en cambio cuenta con un tiempo de aprendizaje y adaptación más amplio, ya que el usuario debe aprender un nuevo modelo de programación y acostumbrar a enfocar y atacar los problemas con esta arquitectura.
- ✓ **Coste:** MapReduce cuenta con un coste bastante menor que una solución MPP estándar. Hay muchas soluciones *open source* que se pueden instalar y ejecutar en infraestructuras de bajo presupuesto mientras que la totalidad de las soluciones MPP son de pago y normalmente se requiere un hardware más costoso (normalmente las soluciones MPP ya vienen con su propia infraestructura, ofrecida por la propia empresa y que acostumbra a tener un precio más elevado).

La valoración teórica final se decanta favorablemente hacia el lado de MapReduce, ya que su diseño está pensado exclusivamente para tratar grandes volúmenes de datos no estructurados en arquitecturas sencillas. Además, tampoco acumula ciertas limitaciones de los sistemas tradicionales. También hay que añadir que el estudio de una arquitectura MapReduce es mucho más enriquecedora, mientras que las MPP están basadas en tecnologías ya conocidas y que a nivel de usuario no aportan tantas novedades.

Por todas estas razones se decidió estudiar las distribuciones Hadoop, ya que es la implementación MapReduce con más soporte y que más éxito ha tenido en el mercado. En el apartado siguiente, se expondrá su funcionamiento y la estructura de la base que aporta el proyecto de Apache, luego se presenta las herramientas de Haddop donde se habla de aquellas que complementan y agrandan las funcionalidades de Hadoop y, finalmente, en el último apartado (*Distribuciones Hadoop)* se estudian y comparan las distintas distribuciones Hadoop que hay en el mercado; con el fin de decidir si se utilizara alguna de las mismas para realizar la fase de pruebas y la implementación del caso expuesto.

La implementación de la arquitectura propuesta se realizará en ambientes virtualizados con el objetivo de simular el procesamiento y llevar a cabo el análisis respectivo de la implementación, igualmente, validar los diferentes complementos de Hadoop para optar por los más adecuados.

4. ANALISIS HERRAMIENTAS HADOOP

Hadoop "Es un framework que permite el procesamiento distribuido de grandes conjuntos de datos a través de grupos de ordenadores que utilizan modelos de programación simple. Está diseñado para detectar y controlar los errores en la capa de aplicación"

Hadoop permite la creación de aplicaciones para procesar grandes volúmenes de información distribuida a través de un modelo de programación sencillo. Está diseñado para ser escalable puesto que trabaja con almacenamiento y procesamiento local (pero distribuido), de manera que funciona tanto para clústeres de un solo nodo como para los que estén formados por miles. Otra característica importante de Hadoop es la detección de errores a nivel de aplicación, pudiendo gestionar los fallos en los distintos nodos y ofreciendo un buen nivel de tolerancia a errores.

El proyecto Hadoop está construido básicamente sobre dos módulos:

- ✓ **Hadoop Distributed File System (HDFS)**: el sistema de ficheros sobre el que se ejecutan la mayoría de las herramientas que conforman el ecosistema Hadoop.
- ✓ **Hadoop MapReduce**: el principal framework de programación para el desarrollo de aplicaciones y algoritmos.

Aparte de estos bloques existen otros proyectos que completan el ecosistema Hadoop para desarrollar soluciones Big Data. En el siguiente apartado se expondrán alguna de ellas.

Actualmente hay dos versiones de Hadoop -1.0 y 2.0- que están siendo usadas por las distintas distribuciones. Ambas versiones tienen diferencias notables en su arquitectura y el hecho de coexistir de momento hace necesario su estudio.

HADOOP 1.0

A pesar de existir ya la versión 2.0 de Hadoop, la primera versión aún es bastante utilizada por muchos desarrolladores al ser la más sólida y estable. El proyecto original de Hadoop se construyó sobre tres bloques fundamentales:

- ✓ **HDFS**
- ✓ **MapReduce**
- ✓ **Hadoop Common**: es un conjunto de las principales librerías y utilidades que la mayoría de proyectos Hadoop utilizan.

4.1. HADOOP 2.0

La segunda versión de Hadoop parte con la base de Hadoop 1.0 añade y modifica algunas características de sus módulos para tratar de resolver algunos de los problemas que tenía y mejorar el rendimiento del sistema. El proyecto Hadoop 2.0 está dividido en cuatro módulos:

- ✓ **Hadoop Common**
- ✓ **Hadoop Distributed File System (HDFS)**
- ✓ **Hadoop YARN**: un *framework* para la gestión de aplicaciones distribuidas y de recursos de sistemas distribuidos.
- ✓ **Hadoop MapReduce**: el sistema de procesamiento principal, que esta vez se ejecuta sobre YARN.

4.1.1. HDFS 2.0

Las principales características de HDFS son:

- ✓ **Sistema de ficheros amigable**. El esquema de HDFS está diseñado para que se parezca lo máximo posible a los sistemas de ficheros conocidos, especialmente al de Unix. Desde el espacio de nombres a los permisos de los ficheros y la seguridad.
- ✓ **Tolerancia a fallos de hardware**. Las instancias de HDFS pueden llegar a tener hasta miles de máquinas trabajando como servidores, almacenando cada una de ellas una parte del sistema de ficheros. Con tal cantidad de componentes formando un sistema, un fallo de hardware que implique la caída o desconexión de uno o más de estos componentes es la regla, no la excepción.
- ✓ **Acceso en streaming**. En las aplicaciones para las que está pensado HDFS, el usuario necesita acceder a los datos con un rendimiento constante y elevado.
- ✓ **Grandes cantidades de datos**. No solo en cuanto a la capacidad total del sistema de ficheros sino también al volumen individual de los ficheros que lo componen. Un tamaño normal y aconsejable para un fichero de HDFS puede ir de los gigabytes a los terabytes.
- ✓ **Modelo simple y coherente**. HDFS está pensado para aplicaciones que necesiten escribir el fichero una sola vez y que, una vez cerrado, no necesite cambios. De esta manera se puede conservar la coherencia de los datos y habilita su acceso rápido.
- ✓ **Portabilidad**. El sistema ha de ser portable a una gran cantidad de plataformas, tanto de hardware como de software.
- ✓ **Escalabilidad simple**. HDFS también permite la fácil expansión del sistema en caliente, pudiendo añadir nuevos nodos sin tener que pausar o parar los procesos que hay en ejecución en el clúster y sin tener que configurarlo; es decir, que el propio sistema se encarga de determinar que bloques de ficheros almacenará y que trabajos realizará.

HDFS tiene una arquitectura de tipo maestro-esclavo. La arquitectura se compone de un servicio único llamado NameNode -que hace la función de maestro- y que se encarga de mantener la coherencia del sistema de ficheros y de permitir el acceso a los ficheros a los diferentes clientes. El otro servicio importante que compone la arquitectura es el DataNode, que acostumbra a estar activo en la mayoría -o todos- los nodos de un clúster. La función del DataNode es la de administrar el almacenamiento de los datos en el nodo donde se ejecutan.

La principal particularidad del sistema de ficheros HDFS es que cada fichero que se almacena en éste se divide en bloques y, a su vez, cada bloque se almacena en un nodo distinto. De esta manera se facilita el uso de modelos de programación como MapReduce, ya que se puede acceder a varios bloques de un mismo fichero de forma paralela. Para asegurar la disponibilidad de los datos y evitar la pérdida de estos debido a un error en alguno de los nodos, cada bloque está, además, replicado en distintos nodos, de manera que la caída de un nodo no implica la pérdida de los datos que contiene.

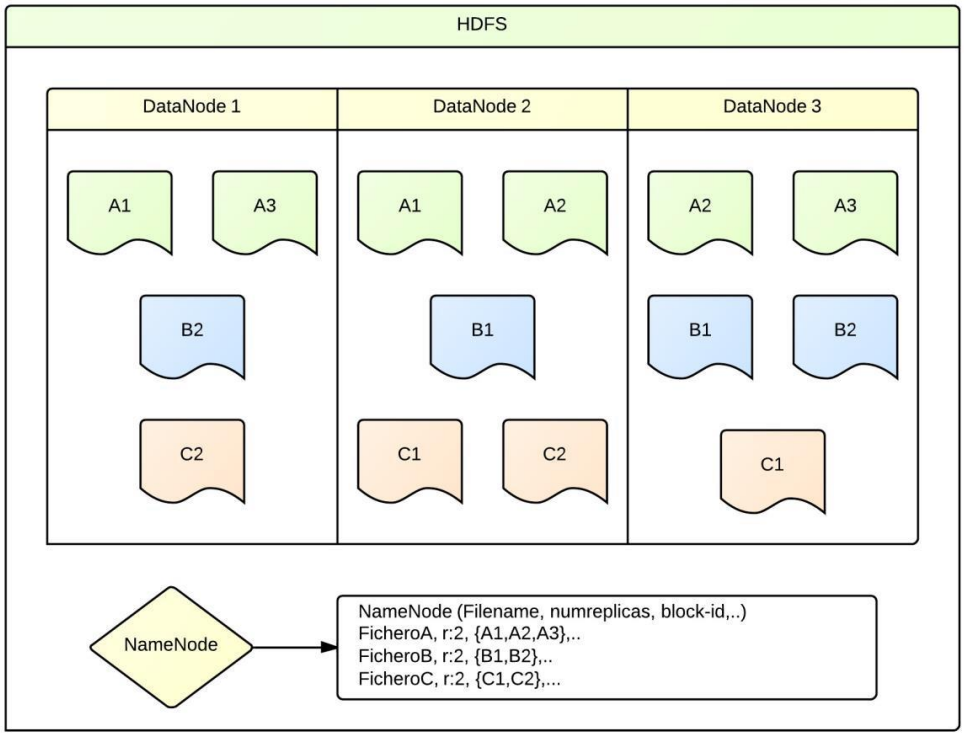


Ilustración 5. HDFS

Hay dos aspectos de la configuración de HDFS importantes en este aspecto:

- ✓ **Tamaño de bloque**: indica el tamaño en bytes o megabytes que debe tener como máximo el bloque de un fichero. Por defecto está configurado a 64 MB o a 128 MB, que son los valores aconsejados para un buen rendimiento del sistema.
- ✓ **Factor de replicación**: el número de veces que se debe replicar cada fichero. Por defecto está configurado a tres y no se recomienda tener un factor de replicación menor. Hay que tener en cuenta que un factor de replicación elevado tampoco ayuda al rendimiento del equipo ya que cada fichero terminaría ocupando más espacio en disco.

En resumen, los DataNode son los encargados de servir los datos que contiene el nodo donde están activos mientras que el NameNode es el que gestiona la coherencia del sistema de ficheros a través de los metadatos. En la Figura anterior se puede observar un ejemplo de la división y replicación de los ficheros en bloques. Se tiene un sistema HDFS con un NameNode y tres DataNodes que contiene tres ficheros: FicheroA, FicheroB y FicheroC; con un factor de replicación dos. Los ficheros se dividen en bloques -cada uno con su propio identificador- y cada bloque está almacenado en dos DataNode distintos. La lista de ficheros y de bloques, con sus identificadores y

en que DataNode está almacenado cada uno, la administra el NameNode. Como se puede observar, cada fichero puede tener su propio factor de replicación (el segundo argumento de los metadatos indica el factor de replicación del fichero).

4.1.2. ALTA DISPONIBILIDAD HDFS

La alta disponibilidad del NameNode se puede conseguir de dos maneras: a través del Quorum Journal Manager o usando Network File System.

QUORUM JOURNAL MANAGER

En este tipo de arquitectura se configura, aparte del NameNode principal, un segundo NameNode que está en modo espera o standby, llamado precisamente Standby NameNode. Este servicio permanece inactivo a la espera de un fallo en el NameNode activo, que es el encargado de realizar las tareas de gestión y administración del sistema.

Para mantener la coherencia de los datos entre los dos NameNodes y mantenerlos sincronizados se crea un grupo de servicios, llamados JournalNodes, cuya función es la de actuar como diarios de todas las operaciones que el NameNode activo va realizando. Este conjunto de JournalNodes se llama Quorum Journal Manager.

El funcionamiento de un sistema HDFS con Quorum Journal Manager es el que se muestra en la *ilustracion 6*. El NameNode comunica a un grupo de JournalNodes (no hace falta que lo haga con todos ya que entre ellos se sincronizan) todos los cambios que se van realizando en el sistema de ficheros -es decir, en los DataNodes-. El Standby NameNode, por su parte, va leyendo el estado del sistema a través de los JournalNodes de manera que cuando se produce un evento de fallida pueda actuar rápidamente como NameNode activo.

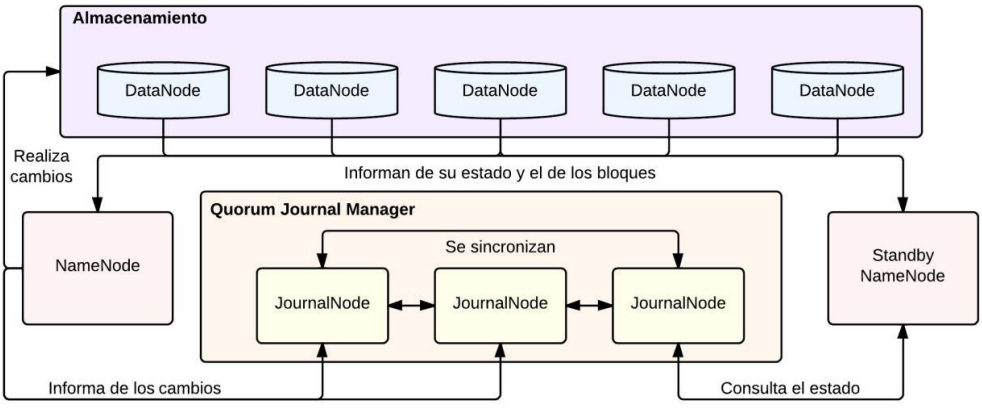


Ilustración 6. Esquema de Servicios HDFS

Para asegurar la coherencia del sistema de ficheros se toman ciertas medidas:

- Sólo puede haber un NameNode activo ya que la existencia de más de uno podría provocar fallos de sincronización y una ruptura del sistema de ficheros. Los JournalNode sólo dan el

permiso para realizar cambios a uno de los NameNodes; de manera que cuando hay un fallo en el activo, el inactivo pasa a reemplazarlo en el rol de servicio activo.

- El NameNode inactivo solamente pasa a activo cuando se ha asegurado de que ha leído todos los cambios contenidos en los JournalNodes.
- Los DataNodes están configurados con las direcciones de los dos servicios de NameNode, tanto del activo y como del inactivo, para enviar constantemente información de los bloques e informar de su estado. Con esto se logra que la transición entre NameNodes sea lo más rápida posible, ya que ambos tienen toda la información necesaria para realizar las tareas de un servicio activo.
- Un JournalNode es un servicio que requiere de pocos recursos por lo que puede ser configurado en nodos con otros servicios del clúster -NameNode, DataNode, JobTracker, etc.
- Se recomienda tener, como mínimo, tres servicios JournalNode para tener un Quorum preparado para la pérdida de uno de los nodos. Para poder aumentar el número de caídas toleradas se debe incrementar el número de servicios en cantidades impares (3, 5, 7…), ya que el Quorum tolera la caída de $(N-1)/2$ nodos, donde *N* es el número total de servicios JournalNode.

El uso del Quorum Journal Manager es totalmente transparente para el usuario y la transición del Standby NameNode a servicio NameNode principal puede ser configurada como automática o manual. Para que esta transición sea automática se requiere de ZooKeeper para detectar los fallos en el NameNode y monitorizar su estado.

NETWORK FILE SYSTEM

Este método para lograr la alta disponibilidad en HDFS también cuenta con un NameNode principal y un Standby NameNode, realizando las mismas funciones que en el caso del Quorum Journal Manager. La diferencia principal es que en lugar de usar un Quorum para la sincronización entre los NameNodes se utiliza un dispositivo de almacenamiento conectado mediante una Network File System. NFS es un protocolo de sistemas de ficheros en red que permite a diferentes ordenadores acceder a ficheros en remoto.

Garantizando el acceso de los NameNodes a este dispositivo a través de NFS, el sistema funciona de manera muy parecida a como lo hace con el Quorum. Cada vez que el NameNode activo realice cambios sobre el sistema de ficheros lo dejará indicado en un fichero de ediciones, almacenado en un directorio del dispositivo NFS. De esta manera el nodo en standby puede ir consultando los cambios realizados en el espacio de nombres del sistema de ficheros. El esquema de trabajo realizado por los nodos sería muy parecido al de la *Figura 2* pero cambiando el Quorum por un dispositivo conectado a través de una red NFS.

Tal y como pasa con el Quorum Journal Manager, solo puede haber un nodo activo a la vez y uno en standby solo cambia de estado a activo cuando está seguro de haber leído todos los cambios del fichero de ediciones.

4.1.3. MAPREDUCE 2.0

MapReduce 2.0 es la capa que más cambios ha sufrido con la segunda versión de Hadoop. Se ha mantenido todas las características que identifican a MapReduce a nivel de usuario: las fases de un proceso; pero se ha renovado por completo su arquitectura y los servicios que la componen.

MapReduce es un proceso batch, creado para el proceso distribuido de los datos. Permite de una forma simple, paralelizar trabajo sobre los grandes volúmenes de datos, como combinar web logs con los datos relacionales de una base de datos OLTP, de esta forma ver como los usuarios interactúan con el website.

El modelo de MapReduce simplifica el procesamiento en paralelo, abstrayéndonos de la complejidad que hay en los sistemas distribuidos. Básicamente las funciones **Map** transforman un conjunto de datos a un número de pares **key/value**. Cada uno de estos elementos se encontrará ordenado por su clave, y la función reduce es usada para combinar los valores (con la misma clave) en un mismo resultado.

Un programa en MapReduce, se suele conocer como Job, la ejecución de un Job empieza cuando el cliente manda la configuración de Job al **JobTracker**, esta configuración especifica las funciones **Map**, **Combine** (shuttle) y **Reduce**, además de la entrada y salida de los datos.

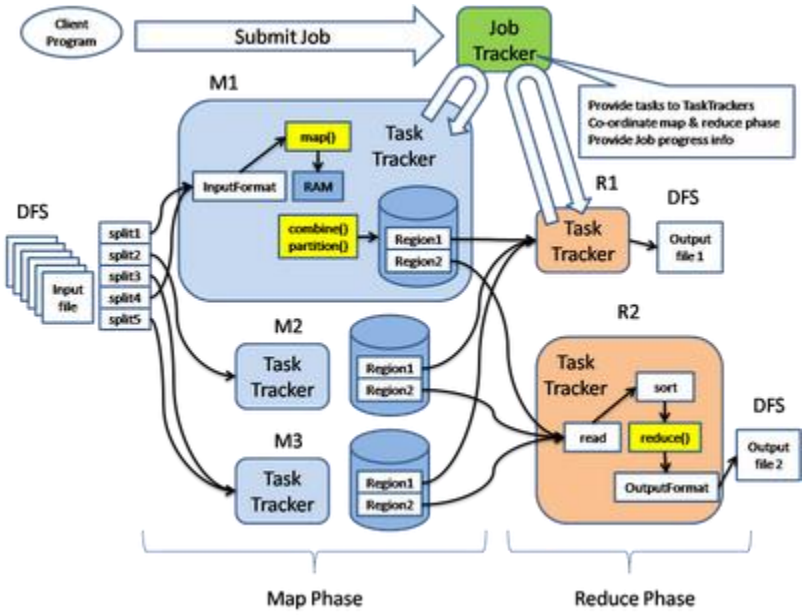


Ilustración 7. MapReduce

4.1.4. ARQUITECTURA YARN

YARN es un motor de gestión de recursos y aplicaciones o procesos distribuidos y es la principal adición de Hadoop 2.0. La principal idea detrás de YARN es la de hacer una gestión más óptima de los recursos de un clúster a la hora de realizar un proceso MapReduce.

Con YARN, el JobTracker deja paso a dos servicios nuevos que se encargan cada uno de una de estas tareas y aparece también el NodeManager. Los servicios trabajan sobre Containers, un concepto abstracto que se utiliza para agrupar los diversos recursos de un sistema (procesadores, memoria, disco, red, entre otros) en una misma noción.

- ✓ **ResourceManager**: es un servicio global para todo el clúster y que se encarga de gestionar los recursos del sistema. Tiene dos partes:
  - **Scheduler**: es el responsable de asignar los recursos a cada aplicación. Está diseñado específicamente para realizar solamente tareas de planificación, por lo que no es encarga de monitorizar ni relanzar aplicaciones caídas. Se encarga de dividir los recursos del clúster a través de diferentes colas, aplicaciones y puede tener varias implementaciones de su política -modificables a modo de *plug-in-*, como el *CapacityScheduler* o el *FairScheduler*.
  - **ApplicationsManager**: se responsabiliza de aceptar las peticiones de creación de trabajos y de crear y asignar un ApplicationMaster a cada una de ellas.
- ✓ **ApplicationMaster**: es un servicio único para cada aplicación -es decir, para cada trabajo MapReduce-. Su principal cometido es el de negociar con el ResourceManager la gestión de los recursos y la de trabajar con los NodeManagers para ejecutar y monitorizar los trabajos.
- ✓ **NodeManager**: es un agente que se ejecuta en cada máquina y es el responsable de los Containers. Se encarga de gestionar los recursos asignados al Container y reportar su estado al Scheduler.

El proceso que siguen estos servicios a la hora de crear un proceso MapReduce se puede ver en la siguiente ilustracion y sigue los siguientes pasos:

1. El usuario ejecuta un proceso y se crea una petición al ApplicationsManager para crear un proceso.
2. El ApplicationsManager habla con el NodeManager para crear un ApplicationMaster.
3. El NodeManager crea el ApplicationMaster.
4. El ApplicationMaster negocia con el Scheduler del ResourceManager los recursos que serán asignados a los Containers.
5. Una vez sabe los recursos, en ApplicationMaster se comunica con los NodeManagers correspondientes para que lancen el proceso MapReduce en un Container con una máquina virtual de Java.
6. El NodeManager lanza y monitoriza el proceso MapReduce sobre un Container.
7. Los Container van reportando su estado al ApplicationMaster, que se encarga de coordinar los procesos.

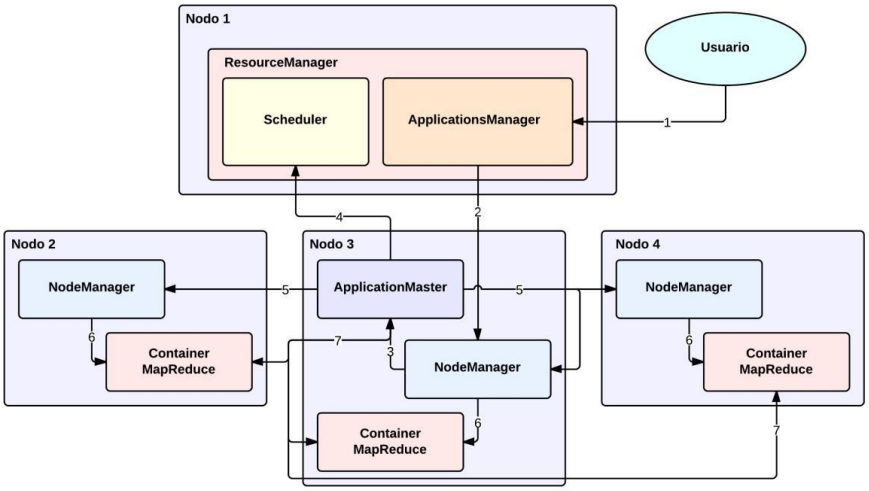


Ilustración 8. MapReduce con YARN

4.2. HERRAMIENTAS HADOOP

Gracias a la creciente comunidad Open Source existen distintos proyectos y herramientas que ofrecen funcionalidades adicionales las cuales son consideradas parte de un gran ecosistema pensado en apoyar las distintas etapas de un proyecto Big Data.

A continuación, se categoriza cada herramienta de acuerdo a la finalidad de cada una, igualmente, se acompaña del link oficial.

4.2.1. CAPTURA Y RECOLECCIÓN DE DATOS – INGESTA DE DATOS

4.2.1.1. Chukwa

http://chukwa.apache.org/

Es un proyecto construido para capturar y analizar grandes volúmenes de datos principalmente logs. Debido a que está construido sobre Hadoop hereda escalabilidad y robustez con el uso de HDFS y Map-Reduce al mismo tiempo que provee adicionalmente un grupo de herramientas flexibles y potentes para visualizar, controlar y analizar los datos.

4.2.1.2. Flume

https://flume.apache.org/

Es una herramienta distribuida para la recolección, agregación y transmisión de grandes volúmenes de datos de diferentes orígenes altamente configurable. Ofrece una arquitectura basada en la transmisión de datos por streaming altamente flexible y configurable pero a la vez simple, de manera que se adapta a distintas situaciones tales como monitorización logs (control de calidad y mejora de la producción), obtención de datos desde las redes sociales (Sentiment Analysis y medición de reputación) o mensajes de correo electrónico.

4.2.1.3. Sqoop

http://sqoop.apache.org/

Es una de las herramientas para potenciar el sistema BI, su funcionalidad permite mover grandes cantidades de datos entre Hadoop y bases de datos relacionales al mismo tiempo que ofrece integración con otros sistemas basados en Hadoop tales como Hive, HBase y Oozie.

4.2.1.4. Uima

https://uima.apache.org/

Otra aplicación interesante con la que podremos analizar grandes volúmenes de datos no estructurados tales como texto, video, datos de audio, imágenes, etc… y obtener conocimiento que sea relevante para el usuario final. Por ejemplo a partir de un fichero plano, poder descubrir que entidades son personas, lugares, organizaciones.

4.2.1.5. Lucene

https://lucene.apache.org/core/

Es una librería escrita en Java diseñada como un motor de búsqueda de textos. Es adecuada para casi cualquier aplicación que requiera la búsqueda de texto completo. Lucene permite indexar cualquier texto o palabra (el texto puede contener letras, enteros, reales, fechas y combinaciones) permitiéndonos después encontrarlos basados en criterios de búsquedas como palabra clave, términos, frases, comodines y muchas más.

4.2.1.6. Apache Kafka

https://kafka.apache.org/

Apache Kafka es un sistema de almacenamiento publicador/subscriptor distribuido, particionado y replicado. Estas características, añadidas a que es muy rápido en lecturas y escrituras lo convierten en una herramienta excelente para comunicar streams de información que se generan a gran velocidad y que deben ser gestionados por uno o varias aplicaciones. Se destacan las siguientes características:

- Funciona como un servicio de mensajería, categoriza los mensajes en **topics**.
- Los procesos que publican se denominan brokers y los subscriptores son los consumidores de los topics.
- Utiliza un protocolo propio basado en TCP y Apache Zookeeper para almacenar el estado de los brokers. Cada broker mantiene un conjunto de particiones (primaria y secundaria) de cada topic.
- Se pueden programar productores/consumidores en diferentes lenguajes: Java, Scala, Python, Ruby, C++ …
- Escalable y tolerante a fallos.
- Se puede utilizar para servicios de mensajería (tipo ActiveMQ o RabbitMQ), procesamiento de streams, web tracking, trazas operacionales, etc.
- Escrito en Scala.
- Creado por LinkedIn.

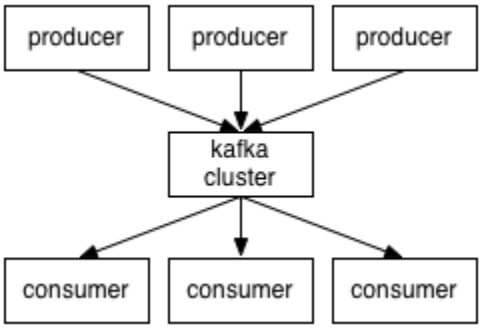


Ilustración 9. Kafka

Kafka tiene como objetivo proporcionar una plataforma unificada, de alto rendimiento y de baja latencia para la manipulación en tiempo real de fuentes de datos. Puede verse como una cola de mensajes, bajo el patrón publicación-suscripción, masivamente escalable concebida como un registro de transacciones distribuidas, la que la vuelve atractiva para las infraestructuras de aplicaciones empresariales.

4.2.1.7. STORM

http://storm.apache.org/

Apache Storm es un sistema que sirve para recuperar streams de datos en tiempo real desde múltiples fuentes de manera distribuida, tolerante a fallos y en alta disponibilidad. Storm está principalmente pensado para trabajar con datos que deben ser analizados en tiempo real, por ejemplo, datos de sensores que se emiten con una alta frecuencia o datos que provengan de las redes sociales donde a veces es importante saber qué se está compartiendo en este momento.

Se compone de dos partes principalmente. La primera es la que se denomina Spout y es la encargada de recoger el flujo de datos de entrada. La segunda se denomina Bolt y es la encargada del procesado o transformación de los datos.

4.2.2. ALMACENAMIENTO

4.2.2.1. Hive

http://hive.apache.org/

Es una herramienta data warehousing que facilita la creación, consulta y administración de grandes volúmenes de datos almacenados en Hadoop. Cuenta con su propio lenguaje derivado del SQL, conocido como Hive QL, el cual permite realizar las consultar sobre los datos utilizando Map-Reduce para poder paralelizar las tareas.

4.2.2.2. HBase

http://hbase.apache.org/

Es la base de datos de Hadoop distribuida y escalable. Su principal uso se encuentra cuando se requieren escrituras/lecturas en tiempo real y acceso aleatorio para grandes conjuntos de datos. Debido a que su base es hadoop adquiere las sus capacidades y funciona sobre HDFS. Puede almacenar en un ambiente distribuido tablas sumamente grandes incluso hablando de billones de registros por millones de columnas, la manera de soportar esta cantidad de datos es debido a que es una base NoSQL de tipo Columnar por lo cual no es posible realizar consultas SQL.

4.2.2.3. Apache Cassandra

http://cassandra.apache.org/

Cassandra es una base de datos NoSQL linealmente escalable. Es distribuida y basada en un modelo de almacenamiento de clave-valor y orientado a columnas. Es totalmente descentralizada, siguiendo una arquitectura de anillo sin nodo maestro o punto único de fallo.

4.2.3. TRATAMIENTO DE DATOS

4.2.3.1. Mahout

http://mahout.apache.org/

Este proyecto permite desarrollar algoritmos escalables de Machine Learning y Data Mining sobre Hadoop. Soporta algoritmos como recomendación, clustering, clasificación y filtro colaborativo, también crear algoritmos para encontrar patrones, que aprendan sobre los datos y que los clasifique una vez termine su fase de aprendizaje.

4.2.3.2. Pig

http://pig.apache.org/

Este proyecto permite analizar grandes volúmenes de datos mediante el uso de su propio leguaje de alto nivel llamado **PigLatin.** Sus inicios fueron en Yahoo donde sus desarrolladores pensaban que el Map-Reduce era de muy bajo nivel y muy rígido por lo cual podías tardar mucho tiempo en la elaboración y manutención. Así pues, nace Pig con su propio lenguaje y trabaja sobre Hadoop traduciendo las consultas del usuario a Map-Reduce sin que éste siquiera lo note. De esta manera provee un entorno fácil de programación convirtiendo las paralelizaciones en dataflows, un concepto mucho más sencillo para el usuario del negocio. Pig tiene dos componentes: su lenguaje PigLatin y su entorno de ejecución.

4.2.3.3. Oozie

http://oozie.apache.org/

Proyecto permite planificar workflows para soluciones que realizan procesos o tareas Hadoop. Al igual que Pig, está orientado al usuario no experto por lo cual le permite definir fácilmente flujos de trabajo complejos sobre los datos.

Oozie funciona como un motor de workflows a modo de servicio que permite lanzar, parar, suspender, retomar y volver a ejecutar una serie de trabajos Hadoop (tales como Java Map-Reduce, Streaming Map-Reduce, Pig, Hive, Sqooq…) basándose en ciertos criterios, como temporales o de disponibilidad de datos. Los flujos de trabajo Oozie son grafos no cíclicos directos (también conocidos como DAGs) donde cada nodo es un trabajo o acción con control de dependencia, es decir, que una acción no puede ejecutarse a menos que la anterior haya terminado.

4.2.3.4. Jaql

https://www.ibm.com/developerworks/jaql

Es un lenguaje de consulta funcional y declarativa que permite la manipulación y procesamiento de datos en formato JSON e incluso semi-estructurados. Fue creado y liberado por IBM bajo Apache License 2.0. Además de ser pensado para trabajar con formato JSON también permite realizar consultas sobre XML, CSV, Archivos Planos y RDBMS.

4.2.4. ADMINISTRACIÓN

4.2.4.1. Zookeeper

https://zookeeper.apache.org/

Es un proyecto de Apache el cual brinda una infraestructura centralizada y servicios que permiten la sincronización del clúster. Zooquiper en pocas palabras se encarga de administrar y gestionar la coordinación entre los distintos procesos de los sistemas distribuidos.

4.2.4.2. Avro

https://avro.apache.org/

Avro, es un sistema de serialización de datos creado por Doug Cutting, el padre de Hadoop.

Debido a que podemos encontrar distintos formatos de datos dentro de Hadoop, Avro se ocupa de que dichos formatos puedan ser procesados por distintos legajes de programación por ejemplo Java, C, C++, Python, Ruby y C#. El formato que utiliza para serializar el JSON gracias a su portabilidad y fácil lectura.

4.2.4.3. Hue

http://gethue.com/

Hue es una herramienta enfocada en los administradores de las distribuciones Hadoop proporcionando una interfaz web para poder trabajar y administrar las distintas herramientas instaladas. Desde aquí puedes cargar o visualizar datos, programar y ejecutar consultas Pig o SQL, realizar búsquedas e incluso programar en pocos pasos un flujo de datos.

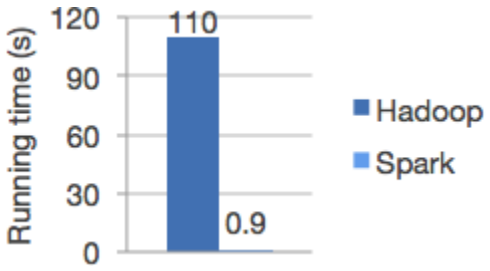
Es una funcionalidad que independientemente de las herramientas que se instalen en algún proyecto Big Data con Hadoop se recomienda incluir.

4.3. Apache Spark

https://spark.apache.org/

Spark ofrece múltiples ventajas con respecto a MapReduce-Hadoop:

- ✓ **Big Data "*in-memory*"**
  Se trata de la cara más visible de Spark. Spark permite realizar trabajos paralelizados totalmente en memoria, lo cual reduce mucho los tiempos de procesamiento. Sobre todo si se trata de unos **procesos iterativos** como los que se usan en el Machine Learning. En la imagen que se muestra a continuación, vemos el benchmark que muestra el rendimiento de Spark respecto a Hadoop-MapReduce.



El hacer trabajos *in memory* no significa que tengamos que comprar servidores con terabytes de RAM (lo cual empieza a alejarse del "***commodity hardware***" que promociona Hadoop). Si algunos

datos no caben en memoria, Spark seguirá trabajando y usará el **disco duro** para volcar los datos que no se necesitan en ese momento. También el programador tiene la posibilidad de definir prioridades, especificando qué datos se tienen que quedar siempre en memoria.

- ✓ **Esquema de computación más flexible que MapReduce**
  Una limitación muy grande de MapReduce es la poca flexibilidad que este paradigma tiene a la hora de crear flujos de datos: solo puedes usar un esquema "Map -> Shuffle -> Reduce". Para optimizar esto, se inventó el **framework** Tez que sustituye el modelo anterior "MapShuffleReduce" por un flujo de ejecución con grafos acíclico dirigido (DAG)

- ✓ **Unificación del mundo streaming-tiempo real y el mundo batch**
  Se trata de otro tema muy importante en el Big Data. Desde hace años se han buscado mecanismos para aportar el componente de tiempo real para Hadoop (framework inicialmente orientado a procesos batchs). Varias soluciones se han propuesto (HBase, Impala, Flume…) pero solo permiten cubrir en parte esta necesidad de tiempo real y de procesamiento en modo stream. Luego llegaron las
  famosas infraestructuras "lambda" y hasta "kappa". Spark simplifica todo esto, en el sentido de que permite trabajar tanto en modo batch como en modo stream-tiempo real. Un mismo framework para unificar 2 mundos.

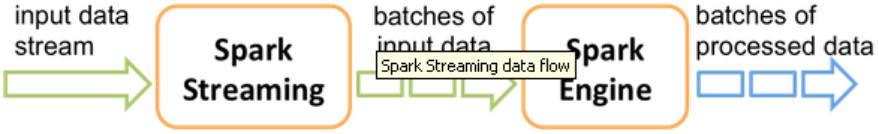


Ilustración 10. Spark Streaming

- ✓ **Una forma de trabajar muy flexible**
  Spark ofrece una API tanto para Java como para Python y Scala. Python y Scala presentan además la ventaja de poder usarse en modo scripting: no se pierde tiempo en iteraciones del tipo "editar un programa, compilarlo, ejecutarlo". Basta con ejecutar los comandos en un intérprete de Spark para poder explotar los datos en nuestro cluster.

- ✓ **Un ecosistema cada vez más completo**
  Al igual que MapReduce en su día, Spark aparece como un framework base en el cual se desarrollan cada vez más aplicaciones avanzadas. Actualmente tenemos:
  - SparkSQL: para explotar los datos con un lenguaje SQL
  - Spark Streaming: Mencionado anteriormente
  - MLib: unas librerías para el Machine Learning
  - GraphX: para la computación de grafos

4.4. BUSQUEDA Y COMPARATIVA DE SOLUCIONES Hadoop MapReduce – SPARK

Hadoop MapReduce es bueno para:

- ✓ **Procesamiento lineal de grandes conjuntos de datos.** Hadoop MapReduce permite el procesamiento paralelo de grandes cantidades de datos. Rompe una porción grande en otras más pequeñas para procesarse por separado en diferentes nodos de datos y recopila automáticamente los resultados en los múltiples nodos para devolver un solo resultado. En caso de que el conjunto de datos resultante sea más grande que la RAM disponible, Hadoop MapReduce puede superar a Spark.
- ✓ **Solución económica**, **si no se esperan resultados inmediatos.** MapReduce es una buena solución si la velocidad de procesamiento no es crítica. Por ejemplo, si el procesamiento de datos se puede realizar durante las horas nocturnas, tiene sentido considerar el uso de Hadoop MapReduce.

Tareas Spark es bueno para:

- ✓ **Procesamiento rápido de datos.** El procesamiento en memoria hace que Spark sea más rápido que Hadoop MapReduce: hasta 100 veces para datos en RAM y hasta 10 veces para datos en almacenamiento.
- ✓ **Procesamiento iterativo.** Si la tarea es procesar los datos una y otra vez. Los conjuntos de datos distribuidos (RDD) resistentes de Spark permiten múltiples operaciones de mapas en la memoria, mientras que Hadoop MapReduce tiene que escribir los resultados provisionales en un disco.
- ✓ **Procesamiento casi en tiempo real.** Si una empresa necesita información inmediata, debe optar por Spark y su procesamiento en memoria.
- ✓ **Procesamiento de grafos.** El modelo computacional de Spark es bueno para los cálculos iterativos que son típicos en el procesamiento de gráficos. Y Apache Spark tiene GraphX, una API para el cálculo gráfico.
- ✓ **Aprendizaje automático**. Spark tiene MLlib, una biblioteca de aprendizaje automático incorporada, mientras que Hadoop necesita un tercero para proporcionarla. MLlib tiene algoritmos listos para usar que también se ejecutan en la memoria. Pero si es necesario, nuestros especialistas en Spark los ajustarán para adaptarlos a sus necesidades.
- ✓ **Unir conjuntos de datos.** Debido a su velocidad, Spark puede crear todas las combinaciones más rápido, aunque Hadoop puede ser mejor si se necesita unir conjuntos de datos muy grandes que requieren una gran cantidad de barajado y clasificación.

Aspectos relavantes para analizar

a) Velocidad. Spark es optimizado para trabajar en memoria, por ende, es mucho más rápido**.**

b) Usabilidad. Una de las cuestiones más habituales al contrastar ambos frameworks está relacionada con su facilidad de uso. En este caso Apache Spark superaría puesto que viene equipado con APIs realmente sencillas para Scala, Python, Java y Spark SQL. Además, aporta feedback en formato REPL sobre los comandos. Por su parte, MapReduce tiene complementos como Pig y Hive que lo hacen algo más fácil de usar, al final lo que sucede es que la lógica

simple necesita más programación (los programas deben estar escritos en Java), por lo que lo que se gana en usabilidad por una parte quedaría perdido por otra.

c) **Rendimiento.** Este punto quizás sea el más complicado de resolver en cualquier comparativa Spark vs Hadoop. La cuestión es que, como ambos procesan los datos de manera diferente, no es nada fácil determinar quién logra un mayor desempeño. Para tomar una decisión habría que tener en cuenta que:

   En lo que respecta a **Spark:**
   - ✓ Trabaja in memory y, por lo tanto, todos los procesos se aceleran.
   - ✓ Pero necesita más memoria para el almacenamiento.
   - ✓ Su rendimiento puede verse mermado debido a la necesidad de utilizar aplicaciones pesadas.

   En el caso de **Hadoop:**
   - ✓ Los datos están en disco y eso hace que todo resulte más lento.
   - ✓ La ventaja es que, en comparación con la otra alternativa, las necesidades de almacenamiento son inferiores.
   - ✓ Al ocuparse de eliminar los datos cuando no son ya necesarios, no produce pérdidas de rendimiento significativas para aplicaciones pesadas.

d) **Seguridad.** Si en usabilidad Spark vencía a Hadoop, en este caso Hadoop no tiene rivales ya que:
   - ✓ Proporciona a sus usuarios todos los beneficios de los avances obtenidos en los proyectos de seguridad de Hadoop (Knox Gateway o Sentry son algunos ejemplos).
   - ✓ HDFS admite la autorización de nivel de servicio, que garantiza los permisos adecuados para los clientes a nivel de archivo
   - ✓ Y, además tiene Hadoop YARN

Por su parte, Spark necesita ejecutarse en HDFS para acceder a permisos de nivel de archivo y, además, para obtener beneficios de seguridad ha de recurrir a Hadoop YARN.

ITEM	Hadoop MapReduce	Spark
Velocidad	X	✓
Usabilidad	X	✓
Rendimiento	✓	✓
Seguridad	✓	X
Total	**2**	**3**

Ilustración 11. Comparativa Hadoop MapReduce vs Spark

Como se puede observar en la comparativa anterior, con un resultado favorable para Spark(3 items a 2 de hadoop mapreduce), sin embargo, Apache Spark no tiene sistema de archivos, por lo tanto, depende del sistema de archivos distribuido de Hadoop, para el presente proyecto se implementara de forma independiente para mejorar el rendimiento y la velocidad de los datos analizados en los logs de eventos de seguridad de los servidores, la cual es otra ventaja de Spark.

5. ARQUITECTURA Y HERRAMIENTAS PROPUESTA

De acuerdo a la problemática expuesta sobre el procesamiento de grandes volúmenes de datos (cerca de 3000 servidores cada uno emitiendo información) recibidos para su posterior análisis, datos referentes a eventos de seguridad, la arquitectura a implementar debe estar orientada de forma relevante al aspecto de recolección en tiempo real, grandes volúmenes de datos, velocidad de análisis, cuya fuente de información no varía (serán archivos planos sobre eventos de seguridad), por lo tanto, teniendo presente las características antes mencionadas del proyecto y las herramientas expuestas en el anterior apartado, se propone la siguiente arquitectura como se muestra en la siguiente imagen:

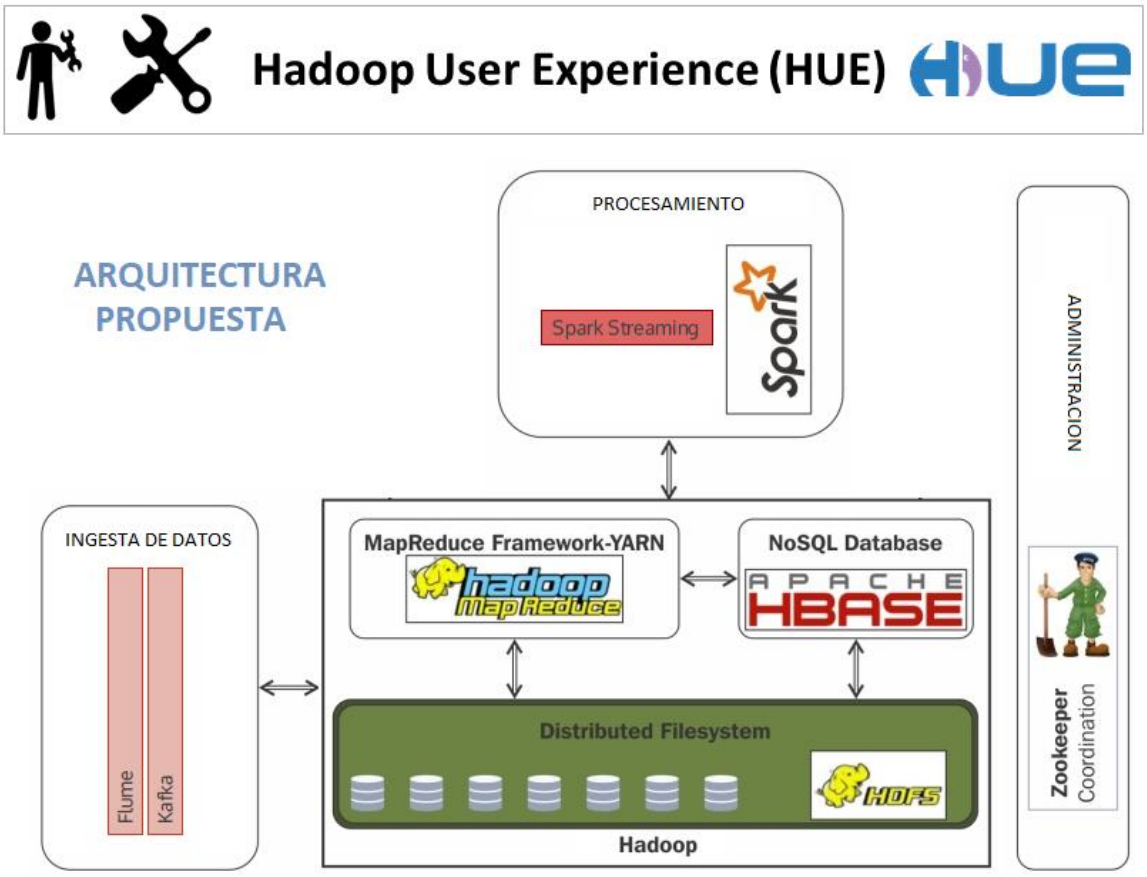


Ilustración 12. Arquitectura propuesta

Por las características mencionadas, de ser en "tiempo real" existen limitaciones que se deben tener presente al momento de realizar las pruebas de concepto pertinentes a la arquitectura planteada:

- **Se debe disponer de suficiente memoria** para almacenar entradas de datos en cola. Si nos fijamos en la diferencia con el paradigma batch, donde los procesos de Map y Reduce podrían ser algo lentos, dado que escribían en disco entre las diferentes fases.
- La **tasa de productividad** del sistema debería ser igual o más rápida a la tasa de entrada de datos. Es decir, que la capacidad de procesamiento del sistema sea más ágil y eficiente que la propia ingesta de datos. Por ello, utilizar spark.

En el proceso planteado los datos que se obtienen en tiempo real van siendo capturados temporalmente para un posterior procesamiento. Ese momento "posterior" es prácticamente instantáneo a efectos de escala temporal. Y es que el "tiempo real", el *streaming*, comienza desde la etapa de ingestión de datos. Tenemos que conectarnos a fuentes de datos en tiempo real, que para el caso analizado son los logs de los eventos de seguridad de los servidores del notariado, para que posteriormente podamos permitir su procesamiento instantáneo.

Siguiendo la metodología de las arquitecturas Bigdata, en los apartados siguientes se presenta el proceso detallado de la arquitectura propuesta.

5.1.1. Origen de datos

Los datos a analizar, los cuales son los elementos de entrada de la arquitectura hacen referencia a registros de log de eventos de seguridad de los servidores de las oficinas del notariado contable, los cuales permitirán analizar amenazas en tiempo real para la ejecución de acciones inmediatas.

5.1.2. Ingesta de datos

Dentro de la arquitectura propuesta, la primera fase es ingesta de datos, esto quiere decir la extracción y transmisión de los datos a analizar, para el caso de uso, los registros de eventos de seguridad, este requisito tiene como aspecto importante la seguridad, integridad, tolerancia a fallos y rendimiento, para este proceso se propone utilizar Kafka, dado que queremos capturar una gran cantidad de información, por lo cual, es conveniente Kafka, con su manejo de colas con un modelo productor - consumidor, el primero captura la información de la fuente(productor) y el segundo dispone la información para ser procesada(consumidor) luego de estar en un topic(tema).

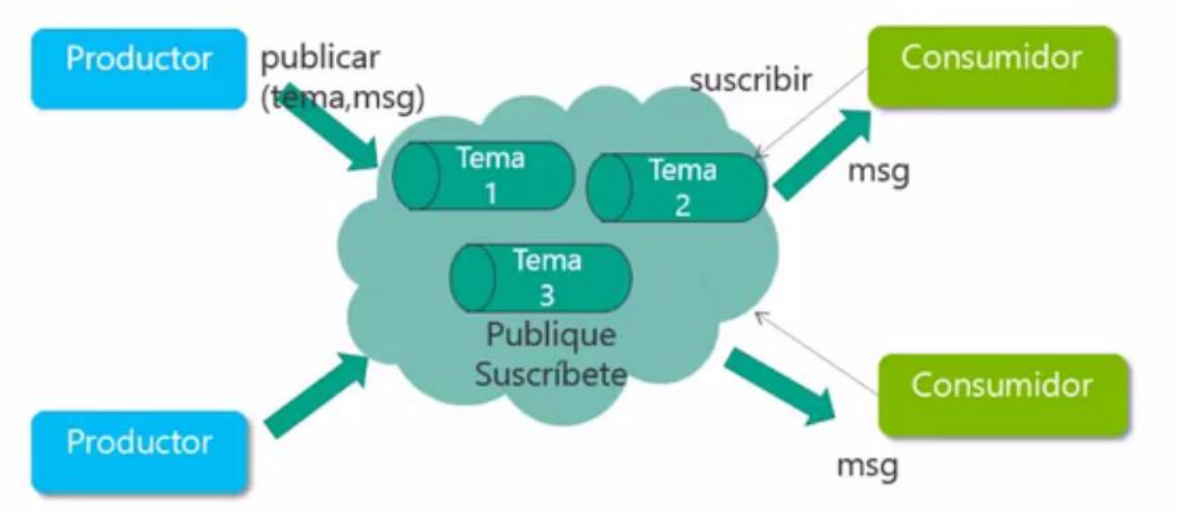


Ilustración 13. Esquema Kafka

Para el caso de uso, los productores de datos son los servidores del notariado, luego de la recolecta se crean los topic (Temas como se muestra en la figura) que están orientados a dejar la

información disponible para ser entregada al procesamiento. Por las características del proyecto es adecuado para escenarios en tiempo real de alto volumen de datos.

Luego de la ingesta y como conector con el procesamiento se recomienda Flume, el cual tiene como finalidad establecer la conexión con el módulo de procesamiento propuesto. Flume debe entregar información al HDFS de Hadoop y al Hbase, para ser procesada por Spark.

5.1.3. Almacenamiento

Para el almacenamiento a largo plazo se proponer utilizar el HDFS de Hadoop junto a la herramienta HBase, esto debido al almacenamiento distribuido en los nodos del clúster. Esto garantiza una menor latencia en la ingesta de los datos. Hadoop HDFS es seleccionado debido a su eficiencia en el almacenamiento distribuido que complementaria a Spark en un clúster y su tolerancia a fallos. Como bien sabemos HDFS gestiona el almacenamiento en el cluster y divide los ficheros en bloques y almacena en copias duplicadas a través de los nodos, se propone que se configure en 3 nodos distintos las copias:

Ejemplo práctico:

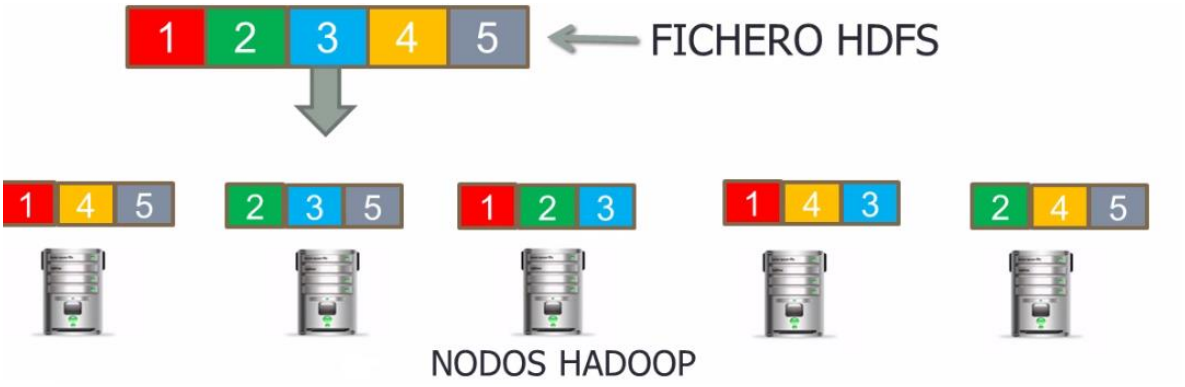


Ilustración 14. HDFS propuesto

Se puede evidenciar en la anterior ilustración, el fichero divido se copia en tres nodos: tolerancia a fallos. Cuando trabajamos con HDFS siempre vamos a tener al momento de crear un cluster Hadoop un nodo Maestro donde se almacena los metadatos, el cual solo tiene información de cómo es el cluster y el resto de nodos son los esclavos, que contienen los datos, por lo cual desde el nodo maestro podemos controlar que se lee y que se escribe en el cluster.

Se recomienda tener presente al momento de crear el cluster, configurar los ficheros core-site.xml (contiene la configuración general del cluster), hdfs-site.xml (contiene la configuración para los datos, para el sistema de ficheros HDFS) y yarn-site.xml (modo de trabajo del proceso Yarn).

Como herramienta de colaboración, tenemos Hbase es tipo NoSQL que se ejecuta sobre HDFS, es tolerante a fallos y permite manejar tablas grandes y masivas de forma sencilla, cuyo tipo de instalación se recomendaría con Fully-Distributed, donde se ejecuten en los distintos nodos del cluster HDFS. También hace uso de Zookeeper que se explicara mas adelante.

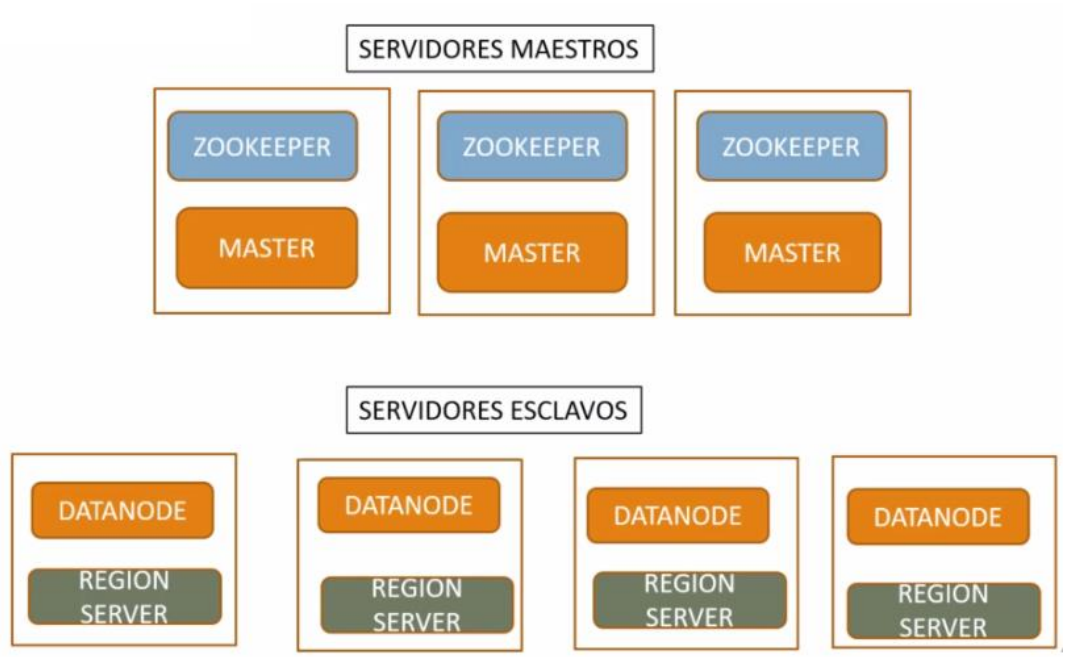


Ilustración 15. HBase

En la ilustración anterior, una zona especial en los servidores esclavos es la Región server, es la encargada de distribuir y gestionar la información y los datos dentro de las tablas.

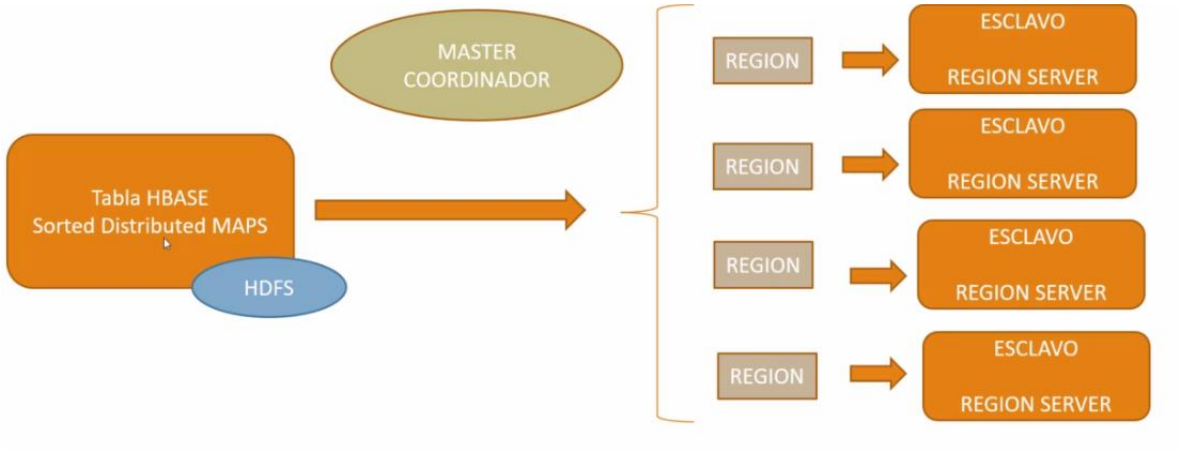


Ilustración 16. Flujo HBase - HDFS

Para el caso de uso es importante el almacenamiento a causa de que se tratan de eventos de seguridad que son fundamentales para la minimización de materialización de amenazas, que conlleven la explotación de vulnerabilidades en los servidores del notariado. De igual forma, los datos almacenados servirán para realizar simulaciones sobre implementación de controles que se opten.

Hasta el presente apartado hemos mencionado la arquitectura que se propone para la ingesta de datos y el almacenamiento de los mismos. A continuación, pasaremos a la sección de procesamiento para finalizar con la visualización.

5.1.4. Tratamiento y procesamiento de datos

De acuerdo a la comparativa realizada en el ítem 4.4 referente a los beneficios que conlleva apache spark frente a Mapreduce, se propone utilizar Spark, debido a sus ventajas como la escalabilidad, alto rendimiento y en especial para el tratamiento de procesos en tiempo real (Spark Streaming), de igual forma, a su rapidez en el procesamiento de grandes cantidades de datos (escritura en base de datos o construir modelos de análisis de datos) y cumplir con el objetivo principal del trabajo de ingesta datos procesados. Como se ha mencionado en varias ocasiones las características del caso de uso en mención son apropiadas para el uso de Spark (Grandes cantidades de Datos y tratamiento en tiempo real). Adicional Spark, tiene el paralelismo que colabora con la eficiencia y rapidez:

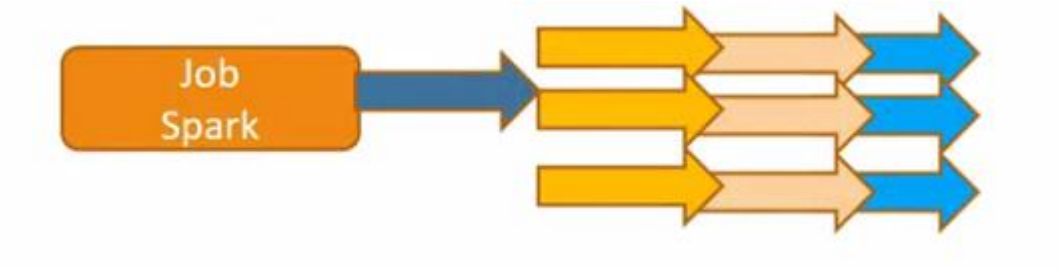


Ilustración 17. Job Spark

Se deben desarrollar los scripts correspondientes para el tratamiento de los datos, estos pueden ser escritos en Scala(nativo), Python, o Java y lanzarlos contra el cluster YARN.

5.1.5. Administración y Visualización

Para la administración se propone Zookeeper, el cual nos sirve para controlar y configurar de forma centralizada entornos distribuidos (mantiene un orden jerárquico), sus datos se almacenan en memoria lo cual lo hace apto para trabajar en escenarios donde el rendimiento es importante, por ello, para el caso de uso actual, se considera su uso. Con zookeper podemos configurar el entorno Hadoop de forma centralizada evitando ese reproceso en cada nodo. Así mismo, lo podemos utilizar para configurar la alta disponibilidad de Hadoop HDFS.

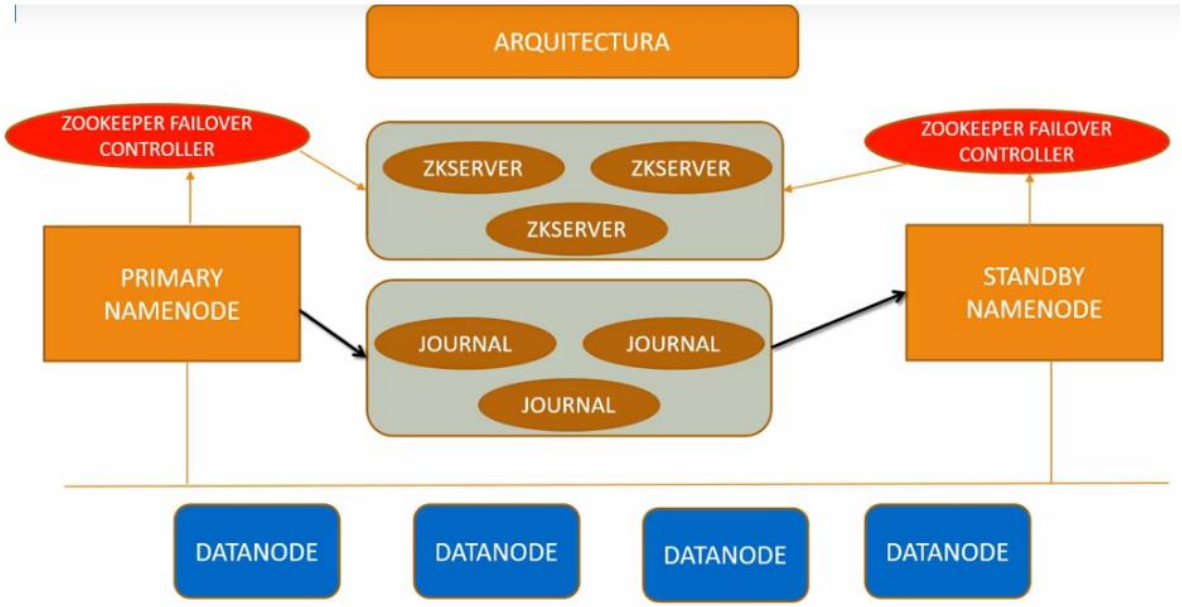


Ilustración 18. Zookeeper

Para la visualización de los análisis mostrados, existen diferentes herramientas que pueden ser utilizadas y desde mi óptica es un punto a variar de acuerdo a lo esperado, para esta propuesta selecciono HUE(opensource) si bien, contiene diferentes funcionalidades y completamente compatible con Hadoop, puede en primera instancia brindarlos diferentes reportes que podríamos usar para tomar decisiones rápidas, así mismo, se puede implementar otra herramienta como es el caso de Kibana, Pentaho u Tableu.

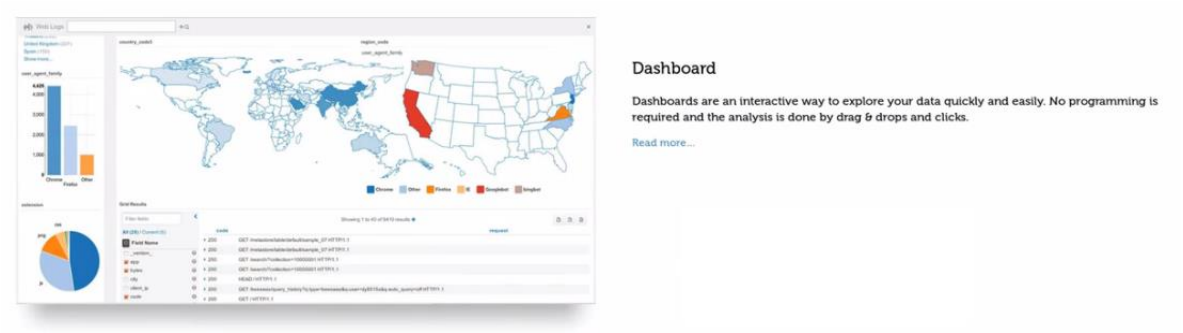


Ilustración 19. HUE

La próxima etapa del proyecto es realizar la prueba de concepto y las configuraciones respectivas, esto con el objetivo de realizar los ajustes necesarios para que la arquitectura genere el resultado esperado, realizando las simulaciones a las que haya lugar y analizando y validando las compatibilidades de las herramientas expuestas como posible solución.

6. SEGURIDAD

La importancia de la seguridad en cualquier tipo de proyecto (tecnológico, arquitectura, minería, construcción, etc.) es importante y es un pilar fundamental, sin embargo, hoy en día existen empresas y "profesionales", una minoría, que piensan que ahorrar en medidas de seguridad harán que sus proyectos sean más económicos y el cliente los elija a ellos sólo por este motivo. No incluir medidas de seguridad en un proyecto (para el presente caso en la arquitectura Big Data) pueden desembocar en problemas graves que pueden incluir pérdida de datos, pérdida de credibilidad pasando por accidentes que no justificarán el ahorro inicial.

Para arquitecturas Big Data, los principales retos de seguridad van desde habilidades de los profesionales, la protección de los sistemas y las implicaciones de vulnerabilidades o ataques.

6.1. Falta de seguridad en el diseño de la solución

El mundo Big Data nos ofrece capacidades útiles para el negocio que no existían. Las diferentes plataformas utilizadas en estos proyectos no siempre cuentan con opciones de cifrado de datos, gestión de políticas, cumplimiento (*compliance*) y gestión de riesgos entre otras características.

6.2. Anonimización

Proteger la información recolectada en la arquitectura Big Data independiente del contenido es imprescindible, con el fin de poder usar los datos con toda garantía y evitar que alguien pueda identificar algo concreto haciendo un análisis profundo.

6.3. Complejidad y diversidad de los datos

La variedad ya sabemos que es una de las V del Big Data junto con Velocidad y Volumen. Contar con numerosas fuentes de datos como archivos, e-mails, aplicaciones, servicios Cloud, datos de dispositivos móviles o IoT hace que tengamos que añadir medidas de protección en cada una de las fuentes y con características adaptadas. Para el caso analizado, la seguridad debe validarse e implementarse en los servidores que generar los eventos de seguridad.

6.4. Pérdida de datos

Prácticamente cada semana nos encontramos con noticias relacionadas con robo de información de servicios en Internet, La pérdida de datos puede desembocar en la quiebra del negocio o la desconfianza en las organizaciones.

6.5. Poca inversión en seguridad

Se ha mencionado anteriormente que por ahorro de costes algunas empresas y profesionales deciden no dedicar la suficiente inversión a implementar medidas de seguridad en sus proyectos. Aunque cada vez más las empresas entienden que la inversión en ciberseguridad debería de ser una obligación, todavía falta un paso más en la concienciación y en la justificación de que la inversión necesaria se realice.

6.6. Falta de habilidades

Al igual que la inversión en ciberseguridad no es siempre la adecuada, sucede lo mismo con la formación en nuevas tecnologías, herramientas y habilidades para los profesionales. No contar con un plan formativo que permita que los empleados añadan nuevas habilidades útiles para el negocio desemboca en fracaso de proyectos principalmente por incumplimiento de costes, plazos o alcance.

Es imprescindible pensar que el atacante siempre va un paso por delante del defensor. En nuestro proyecto tenemos que proteger todos los puntos críticos de la solución que se desarrolle y el atacante sólo tiene que estudiar cuál de esos puntos es el más vulnerable e intentar acceder por él.

6.7. Ruptura de datos

Si el robo de datos es importante no podemos olvidarnos de controlar en todo momento los datos que estamos utilizando para definir nuestra toma de decisiones. Que usemos datos falsos, modificados o incorrectos puede desembocar en una toma de decisiones incorrecta que haga que afecta a nuestro negocio y al del cliente.

6.8. Seguridad en nuestra arquitectura propuesta

Cada herramienta propuesta para las etapas de la arquitectura propuesta en el apartado anterior, tienen su propia seguridad, sin embargo, es realmente conveniente implementar controles de seguridad en las diferentes partes de la arquitectura y se recomiendas las siguientes actividades:

- Diagnóstico de seguridad actual en servidores generadores de eventos. (Fuente)
- Dependiendo del diagnóstico anterior, implementar los controles de seguridad e implementaciones de servicios adicionales de ser necesarios (firewalls, políticas de acceso, definición de perfiles de administración).
- Una vez seleccionada la forma de instalación de la arquitectura de Big Data, en los servidores donde se configuren las diferentes herramientas aplicar controles de seguridad de acceso y políticas de administración, definiendo claramente los roles, perfiles y alcance.

- Se debe asegurar que la transmisión de la información desde la fuente (ingesta de datos) hasta el almacenamiento en la arquitectura propuesta, debe ser cifrada, con algoritmos fuertemente encriptados debido a la criticidad de la información transportada.
- Deben validarse controles de seguridad a nivel del almacenamiento de la información, HDFS, Hbase. Debe revisarse la seguridad e implementar las medidas requeridas y necesarias.
- Para la generación y visualización de los análisis, deben definirse políticas de confidencialidad, conocemos que son eventos de seguridad de los 3000 servidores, por consiguiente, la información es sensible y debe tratarse con la mayor confidencialidad posible.
- Contemplar los desafíos expuestos en el presente apartado, con el objetivo de tenerlos presente a la hora de la implementación.

Las medidas de seguridad nunca son suficientes cuando de tecnología se trate y debe ser una actividad constante la revisión de las mismas, por ellos se considera necesario, que una vez implementada la arquitectura y puesta en producción(dependiendo de la opción seleccionada ver siguiente apartado) debe contemplarse actividades como el análisis periódico de vulnerabilidades, actualización de las herramientas utilizadas, revisión de accesos y realizar los ajustes y/o modificaciones que haya lugar.

7. COMPARATIVA DE IMPLEMENTACIÓN DE LA ARQUITECTURA PROPUESTA BIG DATA - HADOOP

Una vez decidida la arquitectura propuesta para el caso de uso, si bien se va a realizar las pruebas de concepto en modo simulación, es viable, analizar el tipo de solución para implementar en la vida real. Para este análisis se tuvo en cuenta tres tipos de soluciones que se pueden conseguir para Hadoop: una basada en distribuciones, las appliances que ofrecen las compañías y las infraestructuras en cloud como servicio.

7.1.1. DISTRIBUCIONES

Las soluciones basadas en distribuciones no son más que clústeres configurados según las necesidades del usuario para instalarle una distribución Hadoop. El usuario es el encargado de encontrar las especificaciones que deben cumplir los servidores y también de la instalación y configuración de la solución. La principal ventaja de este tipo de soluciones es la flexibilidad a la hora de elegir las especificaciones para que estén dentro de unos límites presupuestarios y que a la vez cumplan con los requisitos del proyecto. También permiten mucha flexibilidad a la hora de realizar modificaciones en la infraestructura y un mayor abanico de opciones a la hora de montar una solución. Por el contrario, requieren de un conocimiento mínimo en administración de sistemas tanto para encontrar las especificaciones adecuadas como para el mantenimiento de las máquinas. También se requiere un conocimiento previo en distribuciones Hadoop para poder realizar la instalación y configuración del software.

7.1.2. APPLIANCE

Las appliances son infraestructuras vendidas por compañías distribuidoras de soluciones Hadoop que también se encargan normalmente de su instalación y servicio técnico. Tienen un coste bastante más elevado ya que también se añade en el paquete el servicio técnico y de soporte anual, además de las licencias y una infraestructura generalmente con unos requisitos muy elevados. Esto beneficia al usuario de las tareas de mantenimiento, pero le resta flexibilidad a la hora de encontrar las especificaciones y también cuando se trata de instalar herramientas o características fuera del appliance, pues acostumbran a ser sistemas muy cerrados.

7.1.3. CLOUD

El auge de las tecnologías de computación en la nube también llega hasta Big Data. Empresas como Amazon, Microsoft o IBM ofrecen sus servidores en la nube para configurar y desplegar soluciones Hadoop. La principal característica de este tipo de solución es que la empresa no se encarga del mantenimiento ni de la localización de las máquinas, ya que solo paga el coste del alquiler de los servidores en lugar de pagar el coste de adquisición (es decir, un valor mensual por el alquiler de la infraestructura). Esto hace que el precio sea más flexible ya que se paga únicamente lo que se usa(demanda), pero para aplicaciones que deban usarse durante un periodo

de tiempo bastante prolongado puede llegar a salir costoso. Además, se pierden los datos almacenados en cuanto se dejan de alquilar los servidores.

Un gran punto a favor es la escalabilidad que ofrecen estos servicios, variar el número de nodos de un clúster suele ser bastante sencillo además de poder escoger que tipo de máquina se desea según las necesidades.

7.1.4. COMPARATIVA

Los puntos valorados en la comparativa entre los tipos de soluciones son los que se han considerado importantes para la realización del proyecto. De esta manera en este punto se valora el tipo de infraestructura que se necesita. Los parámetros escogidos en esta comparativa han sido seleccionados teniendo en mente las capacidades de la empresa y los objetivos del proyecto (Gran Volumen, Velocidad, tiempo real). Las valoraciones van de 1 (peor) a 3 (mejor). A continuación, se explica el significado de cada punto:

- ✓ **Costo**: el precio de la adquisición y mantenimiento de la solución. No se entra a detallar un precio exacto sino el tipo de gasto que implica la adquisición y mantenimiento de la infraestructura. La valoración para este aspecto es:
  - **1**: el precio es elevado y muy poco flexible.
  - **2**: tiene un precio elevado, pero es flexible a las necesidades del proyecto.
  - **3**: el precio es totalmente flexible a las necesidades del proyecto.
- ✓ **Flexibilidad**: las opciones que ofrecen a nivel de software. Significado de las valoraciones:
  - **1**: solo permite trabajar con el software que incluye de serie.
  - **2**: permite trabajar con otro software, pero incluye algún tipo de restricción (permisos, lista de software compatible, etc.).
  - **3**: es totalmente abierto a la configuración de software deseada por el usuario.
- ✓ **Escalabilidad**: las facilidades o dificultades de cada infraestructura a la hora de cambiar el tamaño de los clústeres. La valoración para este aspecto es:
  - **1**: permite la escalabilidad, pero está restringida a un cierto tipo de hardware.
  - **2**: permite añadir y escalar sin importar el hardware, pero requiere un cierto nivel de conocimiento.
  - **3**: añadir o quitar nodos es sencillo, no implica configuraciones nuevas ni trabajar con el hardware.
- ✓ **Seguridad**: controles de acceso, instalación y configuración. La valoración para este aspecto es:
  - **1**: Requiere un alto nivel de seguridad, tanto de hardware como de software, el responsable es quien configura e instala.
  - **2**: La implementación de seguridad, es responsabilidad compartida.

ITEM	Appliance	Cloud	Distribuciones
Costo	1	3	3
Flexibilidad	2	1	3
Escalabilidad	1	3	2
Seguridad	2	2	1
Total	**6**	**9**	**9**

Ilustración 20. Comparativa Implementación de Soluciones BigData.

- ✓ **Costo**: Las appliances son más caras, pero ofrecen simplicidad y rendimiento sin que el usuario tenga que tener grandes conocimientos. En el caso de las infraestructuras en cloud se paga por el uso que se les da, haciéndolas más económicas en usos puntuales, sin perder la simplicidad en la configuración y aportando la ventaja de no tener que adquirir hardware. Las distribuciones ofrecen una solución intermedia, permite hacer la configuración de hardware y de software más adecuada a las necesidades del usuario, de manera que en usos continuos se puede llegar a amortizar la inversión.
- ✓ **Flexibilidad**: en las infraestructuras basadas en distribuciones se permite más margen a la hora de escoger el software que se usa, al tener un entorno de trabajo menos limitado, ya sea en hardware y configuración (como las appliances) o solamente en configuración (como las soluciones cloud).
- ✓ **Escalabilidad**: en las appliances y distribuciones añadir nuevos servidores para aumentar la capacidad de almacenamiento y de procesado implica adquirir nuevas máquinas. En el caso de las appliances es más restrictivo. Para cloud simplemente se piden más servidores sin tener que realizar una gran inversión en tiempo ni en esfuerzo.
- ✓ **Seguridad**: La seguridad juega un papel importante en la implementación de la arquitectura por consiguiente las seguridades deben ser explicitas y trabajadas con mucho cuidado. Por lo tanto, compartir la seguridad puede traer beneficio o no, depende de la declaración de riesgos que se quieran asumir.

Después de realizar las valoraciones oportunas, la infraestructura más adecuada para este proyecto es la basada en cloud, sin embargo, es una valoración personal, donde faltan aspectos que se desconocen actualmente a tener presente a la hora de implementar en la solución real, aspectos como el nivel de conocimiento sobre dichas tecnologías por parte de la empresa, el presupuesto destinado para el proyecto, la comparativa con el costo actual y costo futuro, aquí solo se presenta una comparativa sobre las bondades que ofrece cada alternativa.

8. CONCLUSIONES

En el transcurso de realización de proyecto, se adquirió un conocimiento amplio de tecnologías Big Data y el ecosistema Hadoop, que permitieron cumplir el objetivo planteado creando una propuesta de un diseño de arquitectura Big Data basada en el ecosistema Hadoop para su respectiva implementación a futuro, es un proyecto interesante donde se presentan las siguientes conclusiones del trabajo realizado:

- ✓ Hoy en día para la cantidad masiva de información que es generada se requiere de tecnologías big data que permitan su tratamiento y/o procesamiento, ya sea un procesamiento por lotes o en tiempo real, por ello, se hace necesario conocer todo el ecosistema que encierra el Big Data con sus diferentes herramientas, alternativas y eficiencia. Igualmente, las arquitecturas deben responder a los aspectos V: Volumen, Variedad, Veracidad, Velocidad para que sean completas y funcionales.

- ✓ En el desarrollo del proyecto, se evidencia que para la implementación de una arquitectura Big Data, se requiere de una investigación exhaustiva que encierra desde los conceptos más básicos hasta la aplicación de conceptos complejos, así como también, conocimiento de diferentes áreas de las tecnologías y una fuerte visión de infraestructura.

- ✓ Como se puedo evidenciar durante el desarrollo del proyecto, para generar una solución con arquitectura Big Data, se hace necesario de antemano conocer la problemática minuciosamente y tener la habilidad de seleccionar las tecnologías que mejor se adapte al caso de uso, tener claro los procesos relevantes como conocer la generación de información (las fuentes), que se requiere y que se desea hacer con la información analizada, así como también la forma visualizarla.

- ✓ Para el caso de uso analizado, la información recopilada de los eventos de seguridad de cada servidor del notariado español, será tratada con tecnologías Big Data para realizar su extracción de información, transmisión y posterior tratamiento con el objetivo de generar un análisis en tiempo real de las posibles intrusiones, por ello el principal aspecto que se tuvo en cuenta para la conformación de la arquitectura fue el Streaming.

- ✓ Para el caso de uso expuesto, donde se analizó especialmente el ecosistema Hadoop, unas de las tecnologías seleccionadas es Spark, dentro de la comparativa expuesta en el proyecto (ver numeral 4.4) con MapReduce, Spark es la mejor alternativa para su procesamiento, dado sus ventajas y beneficios, que van orientados al tratamiento de datos en tiempo real con la utilización del paralelismo en su módulo Spark Streaming para cada nodo esclavo.

- ✓ Una vez seleccionado Spark para el procesamiento, se hacía necesario seleccionar el sitio de almacenamiento de archivos, para lo cual se propone trabajar Hadoop HDFS, dado que este presenta beneficios como: una eficiente escalabilidad, haciendo expandir fácilmente el clúster añadiendo nuevos nodos, por su bajo costo de almacenamiento y su tolerancia a fallos.

- ✓ Al momento de la utilización de Kafka, para la implementación de la arquitectura propuesta, se recomienda realizar una planificación detallada sobre escalabilidad a futuro, esto con el objetivo de crear los topics necesarios para la información almacenada. Este es un factor importante en el éxito de esta herramienta.

- ✓ Un aspecto importante que se resalta en el diseño de la arquitectura propuesta, es la tolerancia a fallos, esto debido a que la solución que se requiere por tratarse en procesamiento en tiempo real, debe tener una alta disponibilidad, por lo tanto, cada tecnología elegida para cada proceso presenta esta característica importante. Cabe resaltar que este aspecto es importante para la seguridad y continuidad de la arquitectura, así como también, la confianza en los resultados que genere la misma.

- ✓ Las diferentes arquitecturas Big Data pueden ser implementadas en diferentes escenarios, desde servidores propios hasta servidores en la nube, estos son aspectos importantes que se deben considerar al momento de su implementación, para lo cual aspectos como el presupuesto, administración y escalabilidad juegan un papel importante en la elección que mejor se ajuste a la empresa o compañía. Ver el numeral 6 donde se expone de forma concreta una comparativa que puede ayudar a la implementación a futuro en el ambiente de producción.

- ✓ Durante el desarrollo de propuesto, se tenía previsto poder realizar la prueba concepto y generar un ambiente virtualizado para comprobar los resultados generados, sin embargo, como se comentó en las anteriores conclusiones, Big Data es un mundo extenso y el ecosistema Hadoop comprende una variedad de tecnologías para lo cual la adquisición de conocimiento incurrió en una gran cantidad de tiempo, donde se estudió una gran gama de tecnologías que fueron expuestas en el presente proyecto, por lo cual se concluye que como finalidad del presente proyecto es proponer una arquitectura que pueda ser implementada para la solución del caso de uso.

- ✓ Para el futuro, se propone realizar la prueba de concepto aplicando la arquitectura propuesta en el presente proyecto, detallando cada instalación, configuración e integración de las herramientas seleccionadas, igualmente, evaluando los resultados, eficiencia, agilidad y análisis que genere el modelo propuesto, de igual manera, la tolerancia a fallos y los diferentes planes de continuidad para hacer de la solución una implementación robusta pero eficiente. Se debe tener total atención a la hora de implementar dicha arquitectura, dado que alguna configuración no deseada o el desconocimiento de la tecnología puede generar una solución inadecuada.

- ✓ La seguridad de la información juega un papel importante hoy en día, donde cada vez más, la información está expuesta a diferentes riesgos, sin embargo, no solo se trata de asegurar la información generada, sino también, de generar análisis con la misma para generar estrategias que nos permitan ser más efectivos y agiles para minimizar la materialización de un riesgo, el cual es el caso de uso tratado en el presente proyecto, que se basa de una recopilación de eventos del sistema e infraestructura del notariado español y poder crear

soluciones y tomar decisiones con una mayor rapidez gracias al análisis e implementación de una arquitectura Big Data.

- ✓ Desde mi perspectiva, los avances tecnológicos aparecen constantemente, herramientas surgidas para hacer de cada proceso en las arquitecturas de Big Data sea más eficiente, sin embargo, se recomienda ser prudentes y detenerse a evaluar las necesarias para que se ajusten a la necesidad y no implementar por cuestión de moda. El éxito de implementar una arquitectura tecnológica no está basado en la versión más reciente, es el resultado de un análisis minucioso de la necesidad y el acoplamiento de las tecnologías para la solución respectiva, teniendo presente como pilar fundamental la seguridad de la información y lo que esta conlleva.

- ✓ La seguridad nunca debería de ser un "should" debería de ser un "must". Definir proyectos Big Data sin incluir las normativas, prácticas y componentes de seguridad desde el inicio del proyecto coloca en un riesgo innecesario que afectará a corto o a largo plazo a los beneficiarios del proyecto.

9. GLOSARIO

Alta disponibilidad: es una característica del sistema que asegura una continuidad operacional durante un periodo de tiempo determinado.

API: significa interfaz de programación de aplicaciones. Es una interfaz que define la manera de trabajar con distintos componentes de programación.

Arquitectura maestro-esclavo: arquitectura de red de servidores con una jerarquía centralizada donde el nodo maestro lleva el peso del control mientras que los esclavos realizan el trabajo de procesamiento.

Base de datos: es un conjunto de datos que pertenecen a un mismo contexto y están almacenados en un mismo sistema con la intención de ser usados repetidas veces.

Batch: es una ejecución de una serie de trabajos o procesos informáticos que no requieren de la intervención manual para iniciarse.

Clúster: conjunto de ordenadores interconectados que actúan como si fueran uno solo.

Dataflow: o flujo de información. El diseño de una ejecución de una serie de procesos o autómatas que se comunican enviándose información a través de distintos canales.

Data warehouse: es una colección de datos orientada a un ámbito empresarial u organizativo y almacenada en un sistema no volátil e integrado que ayuda a la toma de decisiones.

Dashboard: o tablero de instrumentos es una interfaz desde la que el usuario puede controlar y administrar una aplicación.

Distribuido: la computación distribuida es un modelo de computación en la que se usan una serie de ordenadores organizados en clústeres.

Escalabilidad: es una propiedad de un sistema que indica su capacidad de reacción y de adaptación a los cambios de envergadura, ya sean al crecer o disminuir.

ETL: proceso de transformación de datos realizado para extraer datos de una fuente y almacenarlos en una base de datos o data warehouse.

Framework: conjunto de conceptos y tecnologías que sirven de base para el desarrollo de aplicaciones. Acostumbra a incluir bibliotecas de software, lenguajes, soportes a aplicaciones de desarrollo, entre otros.

Log: es un fichero que archiva un conjunto de entradas que informan de los distintos eventos - cambios en los estados de los procesos, comunicaciones, errores, entre otros que son producidos en un ordenador.

NoSQL: amplia clase de sistemas de gestión de bases de datos que ofrecen una alternativa al modelo relacional tradicional en aspectos muy importantes, destacando la no utilización de SQL.

Open-source: hace referencia al código distribuido y desarrollado libremente, de manera abierta a todo el mundo, dando acceso al código fuente del proyecto.

Script: archivo de texto plano que contiene una serie simples de comandos -o códigos sencillos en lenguajes interpretados- cuyo objetivo es generalmente el de realizar tareas de orquestación de procesos o monitorización.

SQL: lenguaje de consultas estructuradas para bases de datos relacionales.

Streaming: distribución de datos de manera constante en forma de flujo continuo -sin interrupción-, usada por ejemplo en la transmisión de contenido multimedia a través de internet.

Tiempo real: un sistema en tiempo real es aquel que interactúa de manera dinámica con un entorno generador de datos con el fin de capturar u ofrecer información a medida que se va generando.

Topic: nombre que agrupa un conjunto de mensajes en un sistema Kafka

Workflow: flujo de trabajo que define como tienen que ejecutarse y comunicarse entre ellas diversas aplicaciones o ejecuciones.

10. BIBLIOGRAFIA

[1] https://consulthink.it/es/como-se-construye-una-arquitectura-para-los-big-data-eficiente/ [Visitado el: 01 de noviembre de 2018]

[2] Oguntimilehin, A., & Ademola, E. (2014). A review of big data management, benefits and challenges. Journal of Emerging Trends in Computing and Information Sciences

[3] https://bites.futurespace.es/2017/09/05/como-disenar-una-arquitectura-big-data-y-no-morir-en-el-intento/ [Visitado el: 01 de noviembre de 2018]

[4] https://www.universidadviu.com/arquitectura-big-data-seis-pasos-sacarle-partido/ [Visitado el: 20 de octubre de 2018]

[5] https://johnfaberblog.wordpress.com/2016/07/28/big-data-arquitectura-entorno-hadoop-mapreduce/ [Visitado el: 10 de octubre de 2018]

[6] https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/402826/5-ventajas-de-la-arquitectura-de-Hadoop [Visitado el: 11 de octubre de 2018]

[7] http://es.wikipedia.org/wiki/MapReduce. [Visitado el: 15 de octubre de 2018]

[8] https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/396938/Ventajas-pr-cticas-de-la-arquitectura-de-Hadoop [Visitado el: 10 de octubre de 2018]

[9] http://www.ticout.com/blog/2013/04/02/introduccion-a-hadoop-y-su-ecosistema/ [Visitado el: 20 de octubre de 2018]

[10] https://data-speaks.luca-d3.com/2017/10/el-ecosistema-hadoop-iii-una-gran.html [Visitado el: 19 de octubre de 2018]

[11] http://www.pentahobigdata.com/ecosystem/capabilities/analytics. [Consultado el: 10 de diciembre de 2018]

[12] https://prometeusgs.com/el-ecosistema-hadoop-y-su-impacto-en-la-eficiencia-en-la-gestion-de-datos-masivos/ [Visitado el: 24 de octubre de 2018]

[13] http://blog.auriboxtraining.com/big-data/hadoop-y-su-ecosistema/ [Visitado el: 8 de octubre de 2018]

[14] http://flume.apache.org/ [Visitado el: 27 de octubre de 2018]

[15] http://blog.jacagudelo.com/hadoop-introduccion-componentes-y-ecosistema/ [Visitado el: 12 de octubre de 2018]

[16]http://blog.auriboxtraining.com/big-data/hadoop-y-su-ecosistema/ [Visitado el: 15 de octubre de 2018]

[17]White, T. (2012). Hadoop: The Definitive Guide (Vol. 3). USA: O'reilly Media.

[18]https://es.slideshare.net/Big-Data-Summit/paradigmas-de-procesamiento-en-big-data-arquitecturas-y-tecnologas-aplicadas [Visitado el: 8 de octubre de 2018]

[19]http://www.ticout.com/blog/2013/04/02/introduccion-a-hadoop-y-su-ecosistema/ [Visitado el: 17 de octubre de 2018]

[20]https://data-speaks.luca-d3.com/2017/10/el-ecosistema-hadoop-iii-una-gran.html [Visitado el: 20 de octubre de 2018]

[21]https://prometeusgs.com/el-ecosistema-hadoop-y-su-impacto-en-la-eficiencia-en-la-gestion-de-datos-masivos/ [Visitado el: 21 de octubre de 2018]

[22]http://blog.auriboxtraining.com/big-data/hadoop-y-su-ecosistema/ [Visitado el: 14 de octubre de 2018]

[23]http://www.cs.us.es/~fsancho/ficheros/IAML/BigData.pdf [Visitado el: 3 de octubre de 2018]

[24]https://www.paradigmadigital.com/ecosistema-big-data/ [Visitado el: 6 de octubre de 2018]

[25]Dobre, C., & Xhafa, F. (2014). Parallel programming paradigms and frameworks in big data era. International Journal of Parallel Programming.

[26]http://www.diegocalvo.es/hola-mundo-en-kafka-y-hortonworks/ [Visitado el: 27 de octubre de 2018]

[27]http://www.diegocalvo.es/kafka/ [Visitado el: 27 de octubre de 2018]

[28]http://www.cloudpartnerstm.com/wp-content/uploads/2012/09/Use-Cases-for-Hadoop.pdf. [Consultado el: 7 de octubre de 2018.]

[29]https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/spark-vs-hadoop-quien-saldra-vencedor [Visitado el: 25 de octubre de 2018]

[30]https://www.adictosaltrabajo.com/2014/09/22/introduccion-storm/ [Visitado el: 28 de octubre de 2018]

[31]https://ingenieria.deusto.es/cs/Satellite/ingenieria/es/facultad-ingenieria/big-data-0 [Visitado el: 12 de octubre de 2018]

[32]https://blogs.deusto.es/bigdata/paradigma-tiempo-real-para-sistemas-big-data-ii/ [Visitado el: 12 de octubre de 2018]

[33]https://enmilocalfunciona.io/aprendiendo-apache-kafka-parte-1/ [Visitado el: 03 de Noviembre de 2018]

[34]https://blog.gfi.es/flume-kafka-spark-y-storm-un-nuevo-ejercito-apache/ [Visitado el: 03 de Noviembre de 2018]

[35]http://www.todobi.com/2016/10/caso-de-uso-de-apache-kafka-en-tiempo.html [Visitado el: 03 de Noviembre de 2018]

[36]https://blogs.deusto.es/bigdata/tecnologias-de-ingesta-de-datos-en-proyectos-big-data/ [Visitado el: 26 de octubre de 2018]

[37]http://timeofsoftware.com/map-reduce-con-hadoop/ [Visitado el: 26 de octubre de 2018]

[38]Samandi, Y., Zbakh, M., & Tadonki, C. (2016). Comparative study between Hadoop and Spark based on Hibench nenchmarks. Cloud Computing Technologies and Applications IEEE.

[39]https://spark.apache.org/ [Visitado el: 28 de octubre de 2018]

[40]http://gethue.com/ [Visitado el: 29 de octubre de 2018]

[41]https://avro.apache.org/ [Visitado el: 27 de octubre de 2018]

[42]https://zookeeper.apache.org/ [Visitado el: 27 de octubre de 2018]

[43]https://www.ibm.com/developerworks/jaql [Visitado el: 29 de octubre de 2018]

[44]http://oozie.apache.org/ [Visitado el: 29 de octubre de 2018]

[45]http://pig.apache.org/ [Visitado el: 28 de octubre de 2018]

[46]http://mahout.apache.org/ [Visitado el: 28 de octubre de 2018]

[47]http://cassandra.apache.org/ [Visitado el: 27 de octubre de 2018]

[48]http://hbase.apache.org/ [Visitado el: 28 de octubre de 2018]

[49]http://hive.apache.org/ [Visitado el: 28 de octubre de 2018]

[50]http://storm.apache.org/ [Visitado el: 28 de octubre de 2018]

[51]https://kafka.apache.org/ [Visitado el: 28 de octubre de 2018]

[52]https://lucene.apache.org/core/ [Visitado el: 27 de octubre de 2018]

[53]https://uima.apache.org/ [Visitado el: 27 de octubre de 2018]

[54]http://sqoop.apache.org/ [Visitado el: 27 de octubre de 2018]

[55]https://flume.apache.org/ [Visitado el: 29 de octubre de 2018]

[56]http://chukwa.apache.org/ [Visitado el: 28 de octubre de 2018]

[57]Harness the Power of Big Data. [ed.] Roman Melnyk. s.l.: The McGraw-Hill Companies, 2012.

Esta es para la presentacion
https://es.slideshare.net/slides_eoi/casos-de-uso-del-big-data

Como dijo Benjamin Franklin (1706-1790), político, científico, inventor y uno de los padres de los EE.UU. “La desconfianza y la precaución son los padres de la seguridad” (“Distrust and caution are the parents of Security”).