UNIVERSIDAD DE LA REPÚBLICA

Facultad de Ingeniería

Instituto de Computación

"Big Data enfocado a Decisiones en Tiempo Real y su Aplicación en la Empresa UnoWiFi"

Proyecto de Grado

Emiliano Schiavone | Juan Andrés Olveira

Tutores

Libertad Tansini | Pablo Romero

Usuario Responsable

Gustavo Azambuja - Empresa UnoWiFi

Tribunal

Carlos Testuri | Omar Viera | Adriana Marotta

Agosto 2015

Resumen

Este trabajo de grado, enmarcado en al área de Programación, Bases de Datos e Investigación Operativa, tiene por objetivos un relevamiento del estado del arte de Big Data, así como el análisis de herramientas existentes en la actualidad, enfocadas al procesamiento en tiempo real. El uso práctico relativo a lo estudiado se realizará sobre la realidad planteada por la empresa UnoWiFi teniendo a Gustavo Azambuja como usuario responsable en el transcurso del trabajo.

Nos referimos a Big Data como a la obtención, almacenamiento, análisis y procesamiento de grandes volúmenes de datos para obtener información útil y relevante utilizando herramientas no tradicionales. Este término se ha vuelto muy popular en este último tiempo debido a la cantidad de información generada y el desafío de obtener resultados significativos en base a esos datos.

Siguiendo el objetivo presentado en el primer párrafo, este trabajo obtuvo como resultado un marco teórico amplio de diferentes conceptos relativos a Big Data, así como el análisis de un conjunto herramientas para tratar la información en tiempo real.

Una pieza fundamental en este trabajo es el planteo, análisis y solución de un caso de estudio real brindado por la empresa UnoWiFi. Este caso nos permitió consolidar los conceptos obtenidos en la investigación realizada aplicada a un problema real que retorne resultados valiosos. El mismo está enfocado en el procesamiento en tiempo real de ingentes volúmenes de datos para buscar una solución a la problemática de obtener el conteo de personas que se encuentran en un determinado establecimiento. Dicho valor debe ser obtenido en tiempo real a partir datos recolectados por dispositivos de conexión a Internet en cada establecimiento. Los clientes de UnoWiFi son lugares que ofrecen distintos servicios, tales como restaurantes, shoppings, entre otros, siendo de interés por parte de los mismos saber cuántas personas hay en ellos en un determinado momento.

Para resolver el problema se estudiaron herramientas de Big Data enfocadas al procesamiento en tiempo real. Apache Kafka y Apache Storm fueron las elegidas para llegar a la solución del problema y mostrar cómo dos herramientas de diferentes propósitos pueden trabajar en conjunto para crear una solución.

La metodología utilizada para llegar al resultado consistió en la captura de datos mediante Apache Kafka y procesamiento de los mismos con Apache Storm. El procesamiento se basó en la definición de una serie de filtrados cuya

finalidad fue depurar la información de manera de llegar a un resultado lo más cercano a la realidad posible. Finalmente se realizó un análisis de los resultados obtenidos permitiendo establecer que se obtuvieron valores dentro de los parámetros esperados. Por su parte, el usuario Gustavo Azambuja ratificó los mismos, valorando el aporte a su empresa y posibles aplicaciones futuras.

Palabras claves: Big Data, Tiempo Real, Hadoop, Apache Storm, Apache Kafka, Hortonworks, UnoWiFi.

Índice general

Capítulo 1

Introducción

En los últimos años han sido muchas las definiciones presentadas correspondientes al concepto de «Big Data». Podemos definir Big Data como «conjuntos de datos demasiado amplio para su tratamiento con herramientas tradicionales de gestión de base de datos» [1].

La popularización del término está ligada al documento publicado por McKinsey Global Institute (MGI) en Junio de 2011 [2], donde se define como «conjuntos de datos cuyo tamaño va más allá de la capacidad de captura, almacenado, gestión y análisis de las herramientas de base de datos». Esta definición tiene una clara orientación tecnológica que no refleja la completa realidad en cuanto a su uso por parte de las empresas, cuya intención es aplicar estas tecnologías siguiendo objetivos específicos. El concepto propuesto por Gartner [3] establece lo siguiente:

«Big Data son activos de información caracterizados por su alto volumen, velocidad y variedad, que demandan soluciones innovadoras y eficientes de procesado para la mejora del conocimiento y toma de decisiones en las organizaciones»

En el artículo Critical Questions for Big Data, publicado en Information, Communications, and Society Journal en 2012 [4], se define Big Data como:

«un fenómeno cultural, tecnológico e intelectual que aparece por la interconexión de los siguientes elementos:

1. Tecnología: optimización de la capacidad informática y de la precisión de los algoritmos para recopilar, analizar, enlazar y comparar los datos.

2. Análisis: basarse en desmesurados conjuntos de datos para identificar patrones con el fin de realizar afirmaciones económicas, sociales, técnicas y legales.

3. Mitología: la creencia popular de que los elevados conjuntos de datos ofrecen una forma superior de inteligencia y conocimientos que pueden generar datos

> que anteriormente no eran posibles, con un aura de verdad, objetividad y exactitud».

Nos referimos al concepto Big Data como sistemas informáticos que soporten: gran volumen, velocidad y variedad de los datos, y sean capaces de recolectar, almacenar y procesar la información con el fin de obtener un mayor provecho de la misma.

En esta oportunidad el trabajo buscará acercar al lector a esta creciente corriente, que en estos años ha ido tomando una trascendencia considerable y promete ser un tema de preponderante importancia para los venideros.

1.1. Objetivo general

El objetivo general del proyecto es presentar el concepto de Big Data, realizando el estado del arte, familiarización con las herramientas y resolución de un problema presentado por la empresa UnoWiFi, el cual está dirigido al procesamiento de datos en tiempo real.

1.2. Objetivos específicos

En base al objetivo general planteado, se definen los siguientes objetivos específicos:

- Estudio del marco teórico de Big Data, surgimiento, motivaciones, actualidad y relación con otros paradigmas.
- Relevamiento de herramientas existentes para la construcción de soluciones Big Data profundizando aquellas aplicables a toma de decisiones en tiempo real.
- Uso práctico de una selección de herramientas sobre un caso de uso propuesto por UnoWiFi, enfocado al procesamiento en tiempo real.

1.3. Organización del documento

Este documento se estructura de la siguiente manera. En el capítulo 2 se brindan los conceptos relevantes para comprender el tema de Big Data explicando su origen y ontología del término «Big Data», taxonomía, arquitectura, y tecnologías destacadas en la actualidad. Se resaltará el procesamiento de datos en tiempo real. No solo estaremos informando sino también motivando al lector sobre este nuevo enfoque tecnológico que abre un camino desafiante en el medio informático.

En el capítulo 3 se presenta un ambiente que engloba un conjunto de herramientas y aplicaciones para facilitar el desarrollo de soluciones Big Data llamado Hadoop. Se detallarán herramientas tanto de administración, almacenamiento, recopilación y procesamiento de datos, dando a conocer también distribuciones existentes en el mercado (Hortonworks y Cloudera). Se priorizaron las herramientas Apache Kafka y Apache Storm, las cuales suelen ser utilizadas para el procesamiento en tiempo real. Nos enfocaremos en ellas para la realización del caso de estudio del capítulo 5.

El capítulo 4 contiene lo relativo a la familiarización con las herramientas Apache Kafka y Apache Storm y a la conjunción de ambas para formar una solución. Se presentarán casos de prueba para ambas herramientas con el fin de comprobar el funcionamiento de los conceptos y arquitectura vistas en el capítulo 3. También analizaremos el uso de la distribución Hortonworks, ya que facilita la utilización de las mismas, así como su integración.

En el capítulo 5 se presenta la especificación del caso de estudio, la misión de la empresa UnoWiFi y los objetivos enmarcados en este proyecto. Luego se hará un análisis de la situación actual interpretando los datos obtenidos y cómo procesarlos para obtener un resultado de valor aplicando las herramientas Apache Kafka y Apache Storm.

En el capítulo 6 se muestran las conclusiones globales del proyecto y se presentan posibles lineamientos a seguir en un trabajo a futuro.

Capítulo 2

Estado del arte

En la primera sección de este capítulo se presenta una introducción al tema, explicando el concepto de Big Data. Se definen los conceptos principales y un acercamiento a la arquitectura de Big Data. Presentaremos algunos casos de uso presentes en la actualidad, tanto para desarrolladores como para terceros que desean beneficiarse con el uso de este paradigma. Profundizaremos sobre el procesamiento por lote, en tiempo real y su coexistencia.

2.1. Introducción

Identificar el origen de conceptos relacionados a las Tecnologías de la Información siempre presenta una cierta dificultad y esta no es la excepción. El término «Big Data» fue presentado por primera vez en julio de 1997 en un artículo realizado por los investigadores de la NASA Michael Cox y David Elisworth [5], en el que afirmaron que el ritmo de crecimiento de los datos comenzaba a ser un problema para los sistemas informáticos actuales, denominando a este inconveniente como «el problema del Big Data».

Todos los días se escriben comentarios en Facebook y en Twitter, se suben vídeos a YouTube, los sensores conectados en red recogen ingentes cantidades de datos de los teléfonos móviles, los contadores del gas y la luz, los motores aeronáuticos, las plataformas de perforación y los equipos atmosféricos recaban información que es almacenada. Los satélites registran datos meteorológicos y geográficos, así como información para uso militar. Se crean «datos de desecho» como subproductos de las actividades cotidianas y se almacenan datos de transacciones, por ejemplo los que recogen las cajas de los supermercados, entre otras vías de recolección de información. Todos estos datos son generados minuto a minuto en forma masiva, como podemos ilustrar en la Figura 2.1 la cual representa la gran cantidad de información

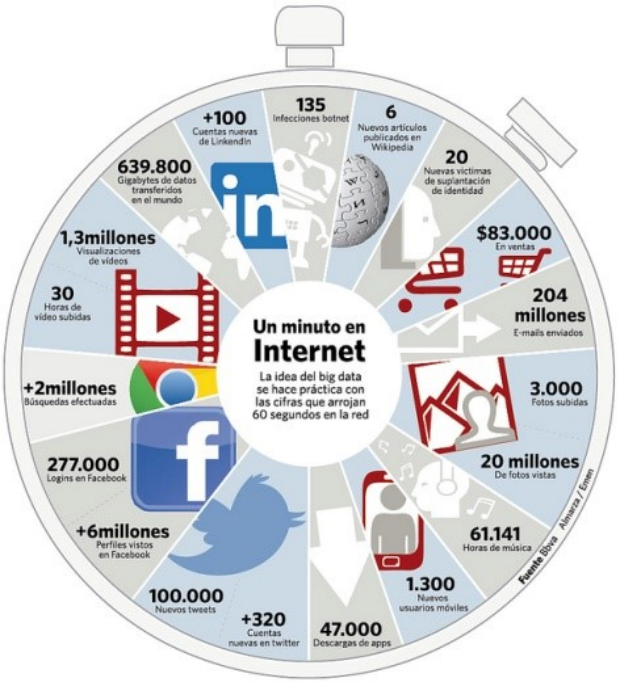


Figura 2.1: Información generada en Internet en un minuto [6]

generada en un minuto en Internet aproximadamente.

> «Más de 2.5 quintillones de bytes (2,5 x 10^{30} bytes) son generados cada día, hasta el punto de que el 90 % de los datos del mundo han sido creados durante los últimos dos años según IBM» [7].

Si grabamos en dvd's todos esos datos y hacemos una columna apilada con los mismos se llegaría a la luna ida y vuelta.

La digitalización ha acelerado el crecimiento de los datos en todas las organizaciones, sectores y economías sin excepción. Empresas, organismos públicos, proveedores de estudios de mercado y meteorólogos están librando una batalla tecnológica para descifrar los conjuntos de datos y extraer parte de su valor. La capacidad para almacenar, consolidar y fusionar información ha hecho que numerosas empresas tecnológicas construyan centros de datos para obtener el mayor provecho a los datos que crece con fuerza [8].

Con esta gran cantidad de datos generados surge la necesidad de poder trabajar con ellos. Es aquí en donde entra en juego Big Data, la cual consiste en la captura, tratamiento, gestión y análisis de enormes cantidades de datos, tan desproporcionadamente vastos que resulta imposible tratarlos con las herramientas de bases de datos y analíticas convencionales.

Las herramientas de bases de datos convencionales conocen el formato de los datos, presentan una estructura para los mismos para posteriormente procesarlos. Cabe recordar que la estructura definida no es una característica específica de los datos manejados en Big Data, por el contrario, carecen de la misma.

En un trabajo publicado por IBM se señala lo siguiente:

> «En términos generales podríamos referirnos como a la tendencia en el avance de la tecnología que ha abierto las puertas hacia un nuevo enfoque de entendimiento y toma de decisiones, la cual es utilizada para describir enormes cantidades de datos (estructurados, no estructurados y semi-estructurados) que tomaría demasiado tiempo y sería bastante costoso cargarlos a un base de datos relacional para su análisis. De tal manera que, el concepto de Big Data aplica para toda aquella información que no puede ser procesada o analizada utilizando procesos o herramientas tradicionales.» [9]

Podemos entender que Big Data se posiciona como el nuevo frente de innovación, competición y productividad. A medida que los formatos digitales se vuelven más sofisticados, se crean más y más datos. Así, un segundo de vídeo en alta definición ocupa 2.000 veces más bytes que una página de texto. La creación de toda esta información está impulsando un rápido avance tecnológico en el terreno del almacenamiento de datos. En 1980, el primer disco duro de un gigabyte, el IBM 3380, tenía el tamaño de un frigorífico, pesaba unos 250 kg y costaba 40.000 dólares. Hoy, se puede conseguir un gigabyte de almacenamiento en una tarjeta de memoria del tamaño de una moneda por unos pocos dólares y con determinadas cuentas de correo electrónico, el almacenamiento en línea es gratuito (hasta cierta cantidad) y existencia de varias plataformas web para este tipo de almacenamiento (Dropbox, GoogleDrive, OneDrive, entre otras).

Según Cisco, en el 2016 la información se multiplicará por cuatro, ya que aumentarán varios factores implicados en las telecomunicaciones, ya sean teléfonos inteligentes, más usuarios en Internet, mayores velocidades de ancho de banda, mejores tecnologías, etc. [10]

En la última jornada de la conferencia que desarrolla anualmente IBM [11] Big Data fue el tema a tratar. En ella se presentó el caso de la empresa de ventas de servicios digitales online, Netflix. La experiencia se refería a la venta de la serie televisiva llamada «House Of Cards» [12]. La empresa basó la decisión de vender este producto digital en los datos que generan los más de 40 millones de usuarios abonados al servicio online. Se utilizó Big Data para conocer al detalle el comportamiento, los

gustos y las preferencias de cada miembro de la audiencia. Netflix puede saber, entre otras cosas, qué género prefieren los usuarios de una ciudad respecto a otra, según la edad, sexo, etc. O qué actor o actriz buscan las mujeres uruguayas de 30 años. Posteriormente intentan cruzar toda esta información, y miles de variables que luego la empresa usa para detectar tendencias, establecer pronósticos y realizar recomendaciones ajustadas a cada usuario [13].

Actualmente la mayor parte de la información fue migrada a la Web para brindar un acceso público. Por ejemplo, el gobierno de Estados Unidos en 2009 generó un sitio web llamado data.gov el cual publica información referida al gobierno (producción, economía, educación, salud, entre otras) [8, 14]. En Uruguay existe un portal similar denominado datos.gub.uy que abre sus puertas a datos abiertos sobre el país [15].

2.2. Conceptos: Las 3 V

Al definir Big Data no solo hay que referirse al gran tamaño de datos, sino que el concepto es más amplio. Se puede realizar una caracterización del concepto de Big Data de acuerdo a las tres V [16]:

- Volumen: un sistema Big Data es capaz de almacenar una gran cantidad de datos mediante infraestructuras escalables y distribuidas. En los sistemas de almacenamiento actuales empiezan a aparecer problemas de rendimiento al tener cantidades de datos del orden de magnitud de petabytes o superiores. Big Data está pensado para trabajar con estos volúmenes de datos.

- Velocidad: una de las características más importantes es el tiempo de procesado y respuesta sobre estos volúmenes de datos, obteniendo resultados en tiempo real y procesándolos en tiempos bastante reducidos. No sólo se trata de procesar sino también de recibir. Hoy en día las fuentes de datos pueden llegar a generar mucha información cada segundo, obligando al sistema receptor a tener capacidad de almacenamiento veloz.

- Variedad: las nuevas fuentes de datos proporcionan nuevos y distintos tipos y formatos de información a los ya conocidos hasta ahora (como datos no estructurados), que un sistema Big Data es capaz de almacenar y procesar sin tener que realizar un pre proceso para estructurar o indexar la información.

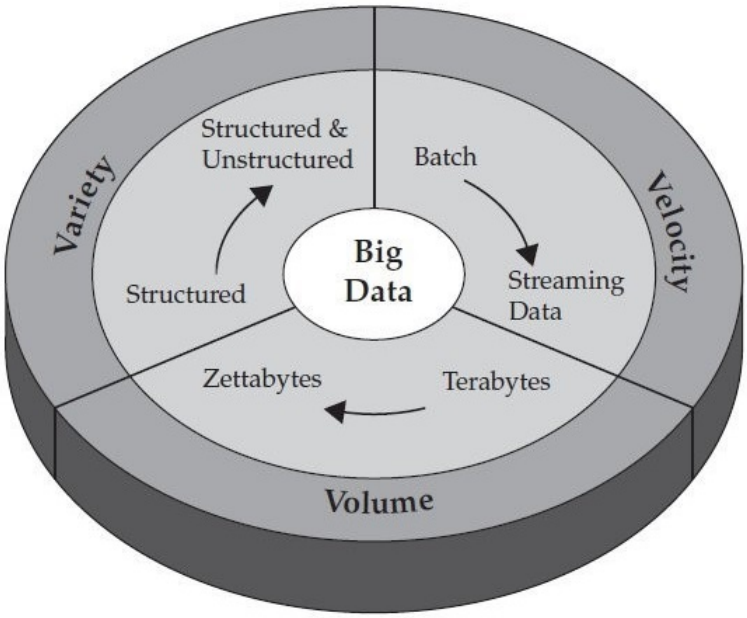


Figura 2.2: Diagrama ilustrativo de las tres V de Big Data [17]

2.3. Tipos de datos

Información acumulada en la nube y otros sistemas no respetan un tipo de estructura general para todos esos datos, por lo que podemos identificar tres tipos de información:

- Datos estructurados: es información ya procesada, filtrada y con un formato estructurado. Es el tipo de datos que más se usa hoy en día

- Datos semi-estructurados: es información procesada y con un formato definido pero no estructurado. De esta manera se puede tener la información definida pero con una estructura variable. Dos ejemplos son las bases de datos basadas en columnas y los ficheros con información en un lenguaje de etiquetas (HTML o XML).

- Datos no estructurados: es información sin procesar y que puede tener cualquier estructura. Se puede encontrar cualquier formato: texto, imagen, vídeo, código, etc. Los directorios de logs de aplicaciones o la información existente en las redes sociales son buenos ejemplos de datos no estructurados.

2.4. Bases de datos no relacionales (NoSQL)

Al momento de considerar las tecnologías requeridas para abordar el problema del Big Data se considera en primera instancia el sistema de gestión de bases de datos a utilizar. La gran mayoría de las bases de datos más conocidas ya se encuentran optimizadas para almacenar y gestionar amplios volúmenes de datos. En los últimos años los sistemas basados en el modelo relacional han tenido un exitoso uso tanto en el ámbito empresarial como en el campo de la investigación. Sin embargo, el espacio que acapara el término Big Data tiene como consecuencia un cambio de paradigma en el modelo de gestión de la información.

En las bases de datos que siguen el modelo relacional se garantizan determinadas propiedades que a priori pueden parecer más que importantes o casi imprescindibles, las mismas constituyen los principios ACID.

Recordemos de forma breve los principios ACID por sus siglas en inglés Atomicity Consistency Isolation Durability, que traducido al español significan Atomicidad, Consistencia, Aislamiento y Durabilidad; es un conjunto de características requeridas para que un grupo de instrucciones puedan ser consideradas una transacción [18]:

1. Atomicidad: cualquier cambio de estado que genera una transacción es único. Esta propiedad asegura que todas las acciones de una transacción se lleven a cabo o ninguna de ellas se realice, por consiguiente en caso de producirse un fallo todos los resultados parciales deberán ser desechos.

2. Consistencia: establece que solo los valores o datos válidos serán escritos en la base de datos. Si por algún motivo una transacción viola esta regla se deberá ejecutar una operación de retroceso dejando a la bases en el estado de consistencia que presentaba antes de iniciar la transacción. Si la misma es ejecutada respetando la propiedad la base de datos pasará a un nuevo estado de consistencia.

3. Aislamiento: asegura que la ejecución de dos o más transacciones sobre la misma información sea independiente y no genere errores.

4. Durabilidad: una vez concluida la transacción, sus resultados son permanentes.

Actualmente resulta imposible gestionar ciertos volúmenes de información sin relajar algunas de estas características. Precisamente de esta relajación y de la necesidad de cumplir con otras propiedades emerge un nuevo paradigma de gestión de datos: NoSQL

Las propiedades que deben cumplir estos nuevos sistemas, principalmente en el entorno de Internet, son:

- Alta disponibilidad
- Tolerancia a fallas
- Gran capacidad de almacenamiento
- Alta capacidad de entrada/salida

Si se quiere cumplir con los puntos anteriores, en general no se cumplirían las garantías ACID, y el sistema presenta una arquitectura distribuida. Además, la interfaz de acceso a la información pierde capacidad de expresión con respecto a SQL dado que la complejidad del esquema de los datos será considerablemente menor.

El teorema CAP (también conocido como teorema de Brewer) [19] señala que para sistemas de almacenamiento distribuidos (en un principio la única solución si se tiene una cantidad de datos elevada), es imposible garantizar las siguientes tres propiedades:

- Consistencia: todos los nodos del sistema distribuido operan sobre el mismo estado de la información, no puede pasar que nodos distintos tengan valores distintos de un mismo dato. Una misma lectura por parte de distintos nodos del cluster deberá arrojar el mismo resultado.
- Disponibilidad: el sistema siempre estará disponible. En caso de que algún cliente realice una operación de escritura o lectura siempre recibirá un mensaje de confirmación, haya fallado o no, pero nunca quedar desconectado del sistema.
- Tolerancia a fallas: el sistema continúa funcionando correctamente incluso cuando alguno de los nodos o parte de la red está experimentando fallas o problemas. El término «partición» hace referencia al hecho de que aunque dos nodos del sistema distribuido hayan quedado desconectados (el sistema ha quedado particionado o dividido), ambos seguirán aceptando lecturas y escrituras aunque eso suponga mantener dos estados diferentes de la base de datos.

En la Figura 2.3 se pueden apreciar estas características con los sistemas correspondientes:

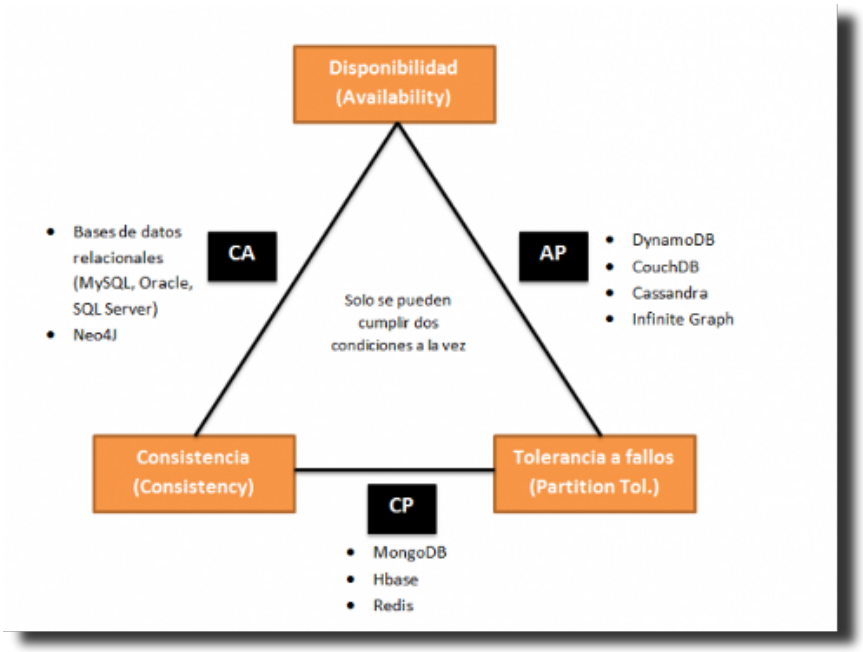


Figura 2.3: Teorema CAP

Dado que un sistema distribuido es la solución más razonable cuando se tiene una gran cantidad de datos para gestionar y que no puede garantizar todas las propiedades a la vez, las bases de datos NoSQL intentan quebrar los principios ACID de las bases de datos relacionales. El fin es poder mejorar otros aspectos necesarios, como lo son la escalabilidad horizontal, simplificación del modelo de datos, capacidad de almacenamiento, mejorar tiempo de respuesta, entre otros.

Como resultado, surge un nuevo conjunto de propiedades, las llamadas «base» (su nombre se debe al símil químico de ácidos y bases). Este concepto hace referencia a las siguientes propiedades:

- Alta disponibilidad: se aplican mecanismos de replicación de datos para que en caso de falla de algún nodo los datos sigan siendo accesibles desde el exterior.
- Estado flexible: se permite a los datos permanecer en estado de inconsistencia, corresponde al programador su posterior manejo.
- Eventualmente consistente: si bien se permite a los datos estar en un estado de inconsistencia, eventualmente se asegura que los datos volverán a un estado de consistencia.

A excepción de las propiedades «base», las bases de datos NoSQL no presentan otras características generales, en contraste con las relacionales que todas tienen objetivos similares entre sí. Cabe señalar que NoSQL no significa que las bases de datos no

utilicen lenguaje SQL, sino que se refiere al término «Not Only SQL», de hecho muchas bases de datos NoSQL permiten el acceso a los datos mediante lenguaje SQL o un grupo de operaciones estándares de ese lenguaje.

Las bases de datos NoSQL se pueden clasificar de acuerdo a cuatro tipos [18]:

1. Clave -Valor: por lo general este tipo de bases de datos guardan los datos en la forma clave-valor, donde la clave es de tipo alfa numérica y el tipo del valor va desde un string a conjuntos de datos más complejos. Este tipo de bases resulta efectivo para las búsquedas rápidas. Por ejemplo se puede usar para la recuperación de valores para tareas relativas a la gestión de perfiles de usuario o recuperación de nombres de productos. Algunas de las empresas que trabajan con estas bases son Amazon (Dynamo) y LinkedIn (Voldemort).

2. Orientada a documentos: como su nombre lo indica están diseñadas para administrar y almacenar documentos. Estos documentos se escriben en lenguajes tales como: XML, JSON, BSON (JSON binario) o similares. A cada documento se le asocia una clave. Se describen a sí mismos por lo que son libres de esquemas. eBay (MongoDB) y BBC (CouchDB) son algunas de las empresas que optaron por este tipo de base.

3. Orientadas a columnas: este tipo de bases están pensadas para realizar consultas y agregaciones sobre amplias cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros. Facebook (Cassandra) y Yahoo (HBase) son organizaciones que trabajan con ellas.

4. Orientadas a grafos: la información se almacena dividiéndola en trozos más básicos y estableciendo relaciones. Estas bases de datos son capaces de obtener rendimientos altísimos en consultas sobre información relacionada, como los contactos de un usuario de LinkedIn o Facebook, o los propios mensajes en twitter (como las búsquedas de retwitts o menciones).

La tecnología NoSQL es utilizada ampliamente por empresas como Google y Amazon. Difiere de las bases de datos relacionales que emplean tablas, esquemas o filas para almacenar datos y después interpretarlos mediante SQL, un lenguaje de consulta estructurado. En lugar de tomar datos, definir relaciones y almacenarlos en una base de datos para su posterior análisis, NoSQL puede analizar los datos en origen. Esto confiere a este lenguaje una ventaja de tiempo real sobre las tecnologías actuales y le hace más capaz de procesar grandes cantidades de datos desestructu-

rados. Por ejemplo, Google analizará las páginas web y documentos buscando una palabra clave en lugar de consultar una base de datos relacional centralizada.

2.5. Arquitectura

La arquitectura del Big Data es similar a un sistema de análisis de información. Se puede dividir en cinco etapas: recolección de datos, almacenamiento, procesamiento, visualización y administración. Para cada una de estas etapas existen herramientas que facilitan su función.

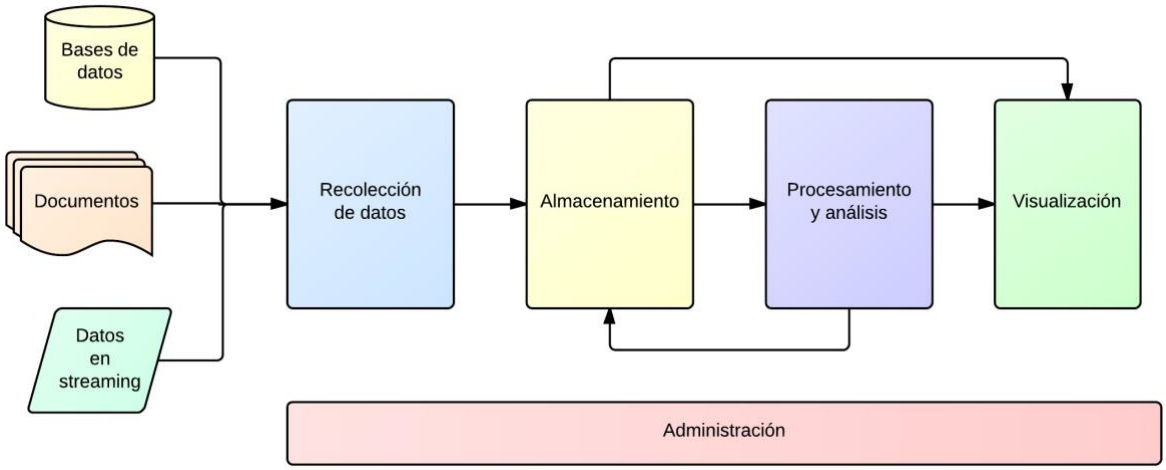


Figura 2.4: Arquitectura de Big Data de cinco etapas

En la figura 2.4 podemos ver el flujo de Big Data, donde la información es recolectada de bases de datos, documentos o datos streaming y luego es almacenada. Luego estos datos pueden ser procesados, analizados y visualizados. Cabe mencionar que las etapas son independientes entre sí, permitiendo generar una solución en cada etapa o realizar una solución que emplee una o varias etapas.

2.5.1. Recolección de datos

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

- 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. Por ejemplo, una herramienta para migrar datos periódicamente de una base de datos a otra.

- Streaming o por transmisión en tiempo real: están conectados de manera continua a la fuente de datos, descargando información cada vez que ésta transmite. En esta etapa, los datos pueden sufrir algún tipo de proceso o cambio si la aplicación así lo requiere, por ejemplo el filtrado de información no deseada o el formato con el que se guardará finalmente en el sistema de almacenamiento. Algunos ejemplos de fuentes de procedencia de estos datos pueden ser redes sociales, ficheros de log, sensores (donde los datos además fluyen con mucha velocidad), o simplemente documentos de texto, audio o vídeo que la empresa ha ido guardando durante bastante tiempo e incluso bases de datos relacionales.

2.5.2. Almacenamiento

En esta capa se encuentran todas las herramientas que permiten almacenar información de gran volumen y de formato variable. El gran tamaño de los conjuntos de datos es la principal razón por la que estas herramientas normalmente son distribuidas y altamente escalables (en caso de llegar al punto de quedar sin espacio disponible, se podría añadir uno varios nodos más al cluster para poder hacer crecer el tamaño total). En la mayoría de los casos estas herramientas permitirán además replicar la información almacenada para tener un almacenamiento tolerante a fallas.

En Big Data existe un abanico de sistemas de almacenamiento suficientemente amplio, debido a la variabilidad de los datos recolectados. Además de varios servicios de almacenamiento en la nube que permiten tener almacenamiento escalable de forma rápida sin la necesidad de comprar, instalar y configurar nuevos servidores.

En esta capa no sólo se almacenan los datos recolectados en la etapa de recolección, sino que además se suelen almacenar los resultados obtenidos al procesar un conjunto de datos en la etapa de procesamiento.

2.5.3. Procesamiento y análisis

En la capa de procesamiento es donde se llevan a cabo todos los análisis y procesamientos de los datos para extraer la información de valor que contengan. Entre la capa de almacenamiento y la capa de procesamiento los datos fluyen en ambas direcciones, ya que se obtiene un conjunto de los datos almacenados para poder procesarlos y analizarlos, o bien ya se ha hecho el procesamiento de los datos y es necesario almacenarlos para poder visualizarlos más adelante o hacer otros procesamientos más complejos que requieren previamente preparar los datos.

Para procesar y analizar los datos se dispone de lenguajes de alto nivel que traducen automáticamente lo que ha expresado por el usuario en procesos complejos, herramientas, paradigmas de procesamiento como MapReduce, paralelismo, etc.

2.5.3.1. MapReduce

MapReduce es un modelo de programación introducido por Google, que ha sido modificado y enriquecido por cientos de colaboradores desde su aparición. Este framework permite la computación paralela sobre grandes colecciones de datos.

Consta básicamente de dos funciones, denominadas «map» y «reduce», elaboradas para que puedan procesar datos de la forma «clave-valor». Los datos de entrada se dividen inicialmente en particiones y se colocan en un sistema de ficheros. En un principio se realiza una etapa de «split» en la que los datos se dividen en partes para luego ser consumidas por la función Map más adecuada de acuerdo a su localización, siendo importante que los datos estén localizados en el mismo nodo en el que se ejecuta la misma [20].

- Map: tiene como objetivo realizar el procesamiento de los datos y dejar los resultados en una lista <clave,valor>, lo que será el mapeo de la información.
- Reduce: combina todas las parejas con la misma clave de la forma que el usuario determina, y guarda los resultados en un fichero del sistema de ficheros compartido por todos los nodos. La unión de los resultados puede corresponder a cualquier tipo de función (agregación, suma, máximo, etc.).

Para fijar ideas, tomemos como ejemplo la idea de contar las apariciones de palabras en un texto dado. En este caso, la función Map tendría como entrada un fragmento de texto, y para cada palabra encontrada escriba en un fichero de salida temporal la pareja clave/valor formada por <palabra>/1. A una misma función Reduce se envían todas las parejas intermedias que contienen la misma clave. Esta procesa las parejas y combina las frecuencias para hacer el cómputo total de apariciones de esa palabra.

Un nodo maestro es el encargado de decidir el número de funciones map y reduce que se ejecutarán, cuántos nodos del cluster intervendrán, y los aspectos de coordinación requeridos para que el proceso sea óptimo.

2.5.4. Visualización

La capa de visualización es la etapa en la que se muestran los resultados de los análisis que se han realizado sobre los datos almacenados. La visualización de los resultados

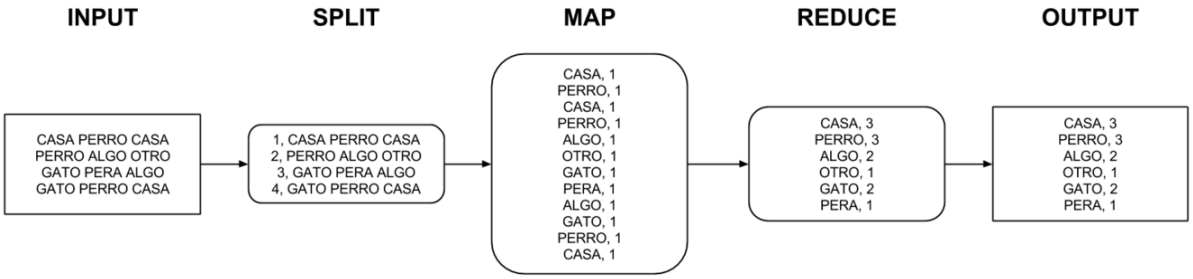


Figura 2.5: Ejemplo Map-Reduce

es normalmente bastante gráfica, de manera que se permita una adquisición rápida de conclusiones para poder decidir cómo actuar o qué estrategias se van a seguir con el objetivo de poder ganar la máxima ventaja o evitar un problema mayor (por ejemplo en la detección de fraude: cuanto antes se detecta una tarjeta robada, menos compras se hacen sin el consentimiento del propietario).

2.5.5. Administración

La capa de administración está presente durante todo el proceso de recolección, almacenamiento, procesamiento y visualización. Esto se debe a que en esta capa se encuentran todas las herramientas que permiten administrar y seguir atento el estado de los sistemas que intervienen durante todos los procesos. Algunos ejemplos de funcionalidades en esta capa son: comprobar que todos los nodos de un sistema distribuido están activos, controlar el factor de replicación o modelo de compresión de los datos almacenados, cargar y ejecutar en el sistema nuevas aplicaciones de análisis de datos, etc.

2.6. Procesamiento por lotes

El procesamiento por lotes (batch) es lo que lleva más tiempo en el mercado, que más se ha usado y que más estable se encuentra. Ha sido la manera más eficiente de procesar información, donde el objetivo es acumular muchos datos, procesarlos y producir resultados por lotes. Este tipo de procesamiento requiere el uso de distintas tecnologías para la entrada de los datos, el procesamiento y la salida.

Se puede decir que Hadoop ha sido la herramienta estrella que ha permitido almacenar gigantes cantidades de datos y escalarlos horizontalmente (es decir agregar más recursos o nodos al cluster), como veremos en la Sección 3.2.

2.7. Procesamiento en tiempo real

En esta sección explicaremos lo relativo al procesamiento en tiempo real. Veremos algunos casos de uso que existen en la actualidad y algunas soluciones amigables diseñadas para que los usuarios puedan utilizarlas.

2.7.1. Introducción

Se refiere al procesamiento y análisis de amplios volúmenes de datos en tiempo real. Hasta hace un tiempo las soluciones Big Data estaban dirigidas al manejo de información estática, donde el procesamiento por lotes se afianzó en el tratamiento de este tipo de problemas. Pero han surgido nuevas necesidades por parte de empresas y organizaciones que han dado lugar a la aparición de nuevas tecnologías dirigidas a contemplar los nuevos requerimientos.

Actualmente el procesamiento en tiempo real de los datos masivos permite mejorar los servicios para analizar el comportamiento de los usuarios y darles respuestas. Las empresas han estado administrando enormes cantidades de datos por años. Ahora las compañías tienen que considerar datos provenientes de más dispositivos, y más información, que requiere de analítica compleja y, por lo menos, casi en tiempo real. Hadoop, referente de la analítica de Big Data, se centra en el procesamiento por lotes. Este modelo es suficiente para muchos casos, pero existen otros modelos de uso en los cuales se requiere información en tiempo real de orígenes altamente dinámicos.

A raíz de esto, se han generado diversas herramientas, adquiriendo un importante papel la denominada Apache Storm (profundizaremos en la sección 3.6). Muchas corporaciones internacionales de gran impacto como Google y Amazon han invertido en este tipo de tecnologías de Big Data.

2.7.2. Casos de uso

Ciertas organizaciones buscan en las tecnologías de Big Data enfocadas al procesamiento en tiempo real la oportunidad de mejorar sus servicios.

Algunas de las áreas en las que aplican son las siguientes:

- Financiera:
    - Gestión de Riesgos / Detección de Fraude: con la crisis financiera de los últimos años, la detección de perfiles de riesgo y el fraude se convirtieron en las principales prioridades.

  - Targeting: la capacidad de comprender mejor a los consumidores, a la perfección en el momento adecuado, permite que una institución financiera pueda optimizar la gestión de relaciones con los clientes rentables y de largo plazo en base a sus necesidades. La adición de una gran cantidad de conocimiento en línea relativamente poco estructurado ofrece una vista mejorada para este fin, lo que podría mejorar la eficacia y la eficiencia de los esfuerzos de marketing. [21]

- Pública:

  - El enfoque Big Data lleva a la predicción de huracanes: modelos estadísticos actuales utilizados para predecir las tormentas tropicales tienen una precisión del 65 por ciento de las veces, lo que resulta en la optimización de recursos. Actualmente los investigadores de la Universidad de Northwestern y sus socios han desarrollado un nuevo método basado en el análisis de datos extensos para el pronóstico actividad de los huracanes de temporada que es hasta un 16 por ciento más exacto que las técnicas anteriores. El nuevo modelo utiliza millones de puntos de datos históricos para predecir la trayectoria y la severidad de las tormentas.[22]

- Telecomunicaciones:

  - Smart Steps es una herramienta que combina tecnologías móviles, geolocalización, sensores, BigData y geo-procesos, que permiten conocer cuántas personas visitan un área en un tiempo determinado, el sexo, la edad, etc. Toda esta información permite realizar estudios estadísticos geotemporales que nos ayudarán a analizar áreas de consumo, hacer comparaciones entre áreas para saber cual tiene mayor potencial para la captación de clientes, adaptar promociones de Street Marketing en sectores con mayor movilidad de personas, determinar cuál es la mejor área para abrir un nuevo local, etc.[23] «Big data es una de las claves fundamentales de la economía digital. Utilizada de forma inteligente y responsable, tiene la posibilidad de transformar los negocios y la sociedad contribuyendo así crecimiento económico y mejorando la vida de las personas» afirma Stephen Shurrock, director comercial de Telefónica Digital [24].

- Ventas:

  - En 2004, Wal-Mart reveló que tenía más de 460 terabytes de datos almacenados. Y utilizan sus datos para impulsar prácticamente cada decisión

que tomen acerca de su negocio: almacenamiento de productos en base a la demanda esperada en un punto específico en el tiempo (basado en una mirada de factores tales como día, hora, clima, eventos especiales, etc.), compra de productos a proveedores, el comportamiento de compra de los clientes (no sólo en lo que los productos son comprados, sino también cuales componen una compra básica).

Wal-Mart está observando todo lo que ocurre en su negocio lo más cercano al tiempo real ya que son capaces de responder de forma proactiva a eventos y poner a la venta ciertos productos antes que nadie. Por ejemplo, cuando ocurrió el huracán Charlie, se establecieron los patrones de compra justo antes del paso del huracán, descubrieron que las ventas Strawberry Pop-Tart aumentaron aproximadamente 7 veces, junto a la cerveza fueron de los productos que registraron mayor venta. Antes de que ocurriera un próximo huracán, se aseguraron de que todas las tiendas en su camino fueran abastecidos con ese producto y cerveza. [25]

2.7.3. Soluciones

En esta subsección mostramos algunas de las soluciones que han implementado organizaciones de gran envergadura como son Google y Amazon, apoyándose en el procesamiento en tiempo real a través de plataformas web. Estas soluciones tienen la ventaja de que si no se dispone de hardware para almacenamiento y procesamiento de grandes volúmenes, una opción viable es la ofrecida por estas empresas. Una de sus desventajas es que los datos a usar estarán compartidos con la empresa que brinda el servicio.

2.7.3.1. Amazon Kinesis

Amazon Kinesis es un servicio totalmente gestionado para el procesamiento de transmisiones de datos en tiempo real a una escala masiva. Permite escribir con facilidad aplicaciones que procesen información en tiempo real, desde fuentes como visitas de sitios web, información de marketing y financiera, instrumentos de fabricación y redes sociales, así como datos de medición y registros operativos.

Con las aplicaciones de Amazon Kinesis, se puede compilar paneles en tiempo real, capturar excepciones y generar alertas, gestionar recomendaciones y tomar otras decisiones empresariales u operativas en tiempo real. También puede enviar datos fácilmente a una variedad de servicios diferentes como Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB o Amazon Redshift [26].

2.7.3.2. Google Cloud DataFlow

Google ha lanzado una plataforma para gestionar datos como servicios. Se complementa con la tecnología Google Query y hace que el Big Data sea accesible para cualquiera. Greg DeMichillie, director de gerencia de producto para el equipo de nube de Google, afirmó que el anuncio de Cloud Dataflow durante la presentación en Google I/O es solo el inicio de sus actividades para Big Data. Una de las declaraciones que más están resonando es que el nuevo esquema de Dataflow es el sucesor de MapReduce dentro de la compañía [27].

Fueron los mismos de Google los que pusieron en marcha, ya hace unos diez años, el algoritmo MapReduce y el entonces llamado Google Distributed File System, que pasaría después a conocerse con el nombre de HDFS, tras la creación de Hadoop por parte de Doug Cutting y Michael Cafarella.

Cloud Dataflow permitirá a los desarrolladores crear pipelines de datos, facilitará su almacenamiento, transformación y análisis. El servicio que se ofrece puede realizarse en tiempo real o trabajando por lotes (batch) en el sistema. Por el momento la versión beta es privada y por tanto no será fácil acceder al servicio y comprobar sus mejoras respecto a MapReduce.

El enfoque según la compañía del motor de búsqueda es «contribuir a que los usuarios puedan obtener valor de sus datos al tiempo que disminuyen sus costes operativos sin tener los inconvenientes de tener que desplegar, mantener y escalar con una infraestructura» [27].

2.7.3.3. Google Big Query

La consulta de conjuntos de datos de gran envergadura puede llevar mucho tiempo y tener un costo alto sin el hardware y la infraestructura adecuados. Google BigQuery resuelve este problema al permitir realizar consultas tipo SQL veloces en tablas de solo anexión con la potencia de procesamiento que ofrece la infraestructura de Google [28].

Para poder consultar los datos, primero se tienen que cargar en BigQuery. Se pueden cargar los datos de forma masiva mediante una tarea o transmitir registros individualmente.

Las consultas se escriben en el dialecto SQL de BigQuery. Admite tanto el método de consulta síncrona como el método de consulta asíncrona. Ambos están controlados por una tarea, pero el método síncrono incluye un valor de tiempo de espera hasta que la tarea haya finalizado para ofrecer un resultado.

Existen funciones para poder integrar las consultas a la aplicación que se esté desarrollando y no tiene por qué ser una solución Big Data.

2.7.3.4. DataSift

Proveedor de flujos de datos en tiempo real. Da la posibilidad de conectar un sistema y recibir eventos de interés de Twitter firehose, Facebook, YouTube, Wikipedia, entre otros en tiempo real. DataSift fue fundada por Nick Halstead para ayudar a las organizaciones a mejorar su comprensión y uso de los medios sociales, se centra en la producción de la tecnología de filtrado de los datos del estado de la técnica y de impulsar la innovación en Big Data. DataSift comenzó su vida como una rama de tweetmeme, el servicio de Twitter feed de noticias altamente popular. Tras ser testigo de lo rápido que impactó socialmente, Nick Halstead se propuso crear una plataforma para ayudar a las empresas a administrar medios de comunicación social y aprovechar los conocimientos que se encuentran dentro de los datos.[29]

Su objetivo, como el de muchos otros servicios que existen en el mercado es desarrollar una herramienta que consiga extraer valor de la ingente e ingobernable cantidad de información en tiempo real con que somos bombardeados cada día. Ayudarnos a filtrar esos torrentes de datos, reducir el «ruido» y convertirlos en información relevante, útil y valiosa. [30]

El servicio ofrece la posibilidad de crear nuestros propios streams de información, tomando como fuentes o inputs cualquier servicio social (Twitter, Facebook, Google Buzz, MySpace, WordPress, Digg, SixAppart) o los feeds de cualquier blog o página web. Ahora incluso se pueden añadir a streams los datos de terceras partes (servicios como Salience, TweetMeme, Peer Index, Klout o InfoChimps).

Además, DataSift también permite establecer filtros personalizados sobre esos feeds de noticias que se utilizan como inputs para los streams. Filtros en función de una gran variedad de variables como por ejemplo la localización, las sensaciones (menciones negativas o positivas) o el tipo de contenido.

Es escalable, la plataforma fue diseñada desde cero para aprovechar el crecimiento exponencial de los datos sociales. Se está continuamente escalando para mantener el procesamiento de amplias cantidades de datos en tiempo real y las interacciones de archivo en el almacén de datos de la plataforma.

Control total de plena confianza, gestionan toda la infraestructura. Eso da un control total sobre todas las políticas de seguridad en todo nivel. Datos certificados, los centros de datos cumplen con certificaciones como ISO 27001 E ISO 3001. Trabajan con clientes empresariales exigentes para ofrecerles soluciones robustas [31].

La confiabilidad de Datasift se utiliza en todo el mundo por muchos clientes empresariales para aplicaciones de negocio. Brindan la infraestructura los equipos a todo horario. La plataforma opera con gran nivel de redundancia, por lo que hace frente al intento de caída en el servicio

Hay algunos medios de recolección de datos los cuales son gratis y otros pagos.

En la figura 2.6 se muestra un resultado utilizando esta plataforma, donde hemos creado un stream que recopile todos los mensajes de Facebook públicos que contengan la palabra «Hello» y en tiempo real nos muestra cuales son esos mensajes.

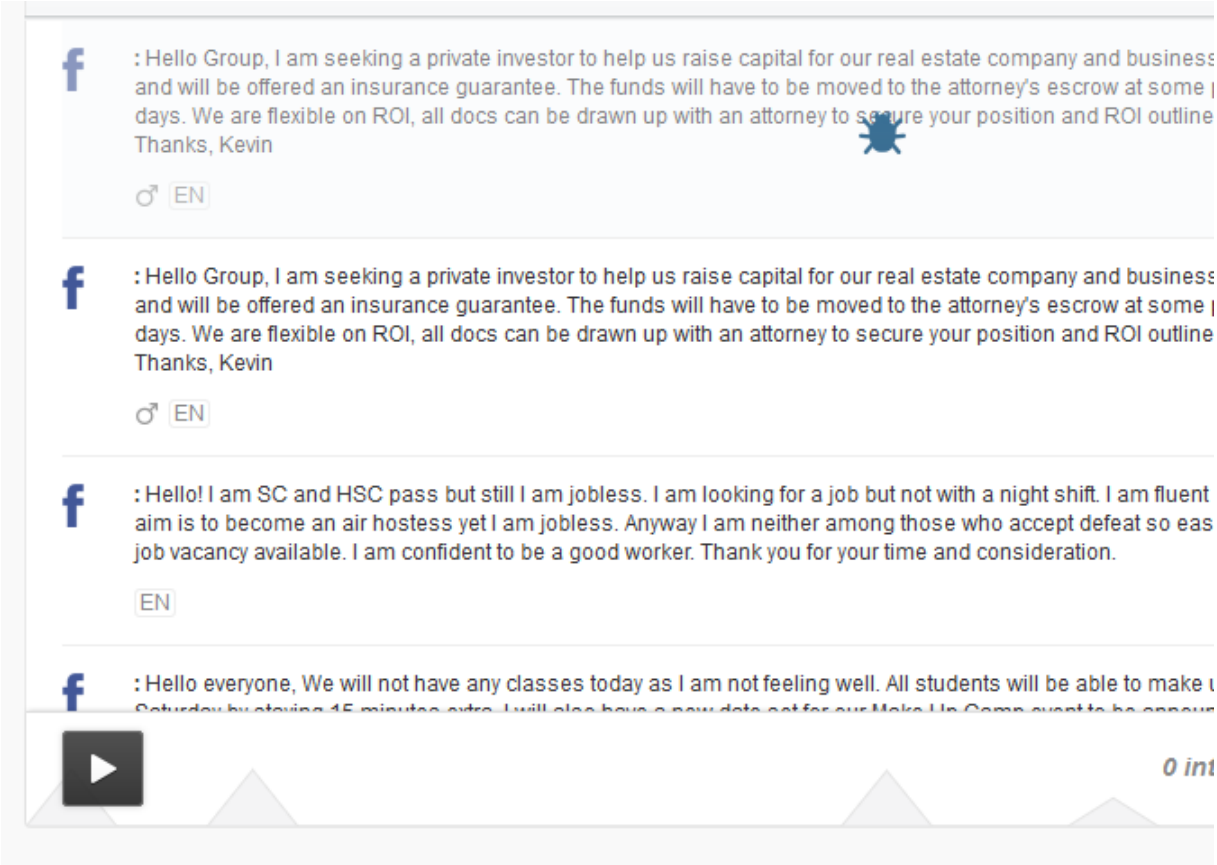


Figura 2.6: Resultado del uso de la plataforma Datasift

2.8. Arquitectura Lambda

La Arquitectura Lambda fue propuesta por Nathan Marz, uno de los referentes en lo que a Big Data se refiere. El objetivo principal de esta arquitectura es brindar una estructura integral para permitir el procesamiento tanto de datos estáticos co-

mo aquellos que están en movimiento, la unión del procesamiento por lotes con el procesamiento en tiempo real [32].

Esta arquitectura se divide en tres capas:

- Capa Batch
- Capa de Servicio
- Capa de Velocidad

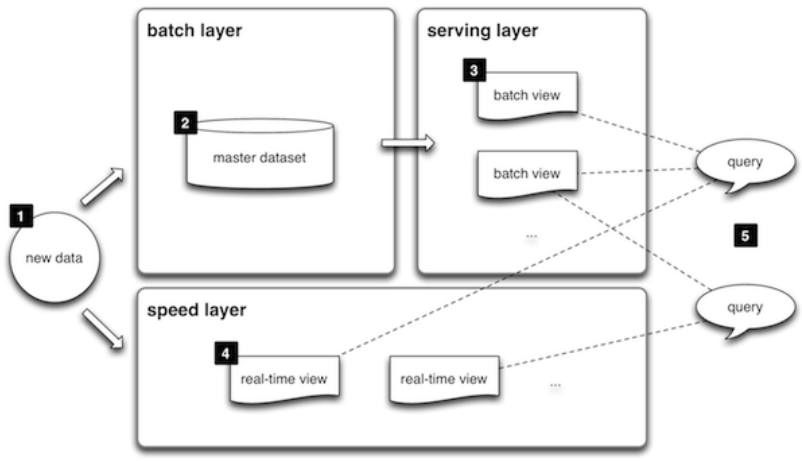


Figura 2.7: Diagrama de arquitectura lambda [33]

2.8.1. Capa Batch

La capa de procesado batch es responsable del almacenamiento de los datos en un conjunto de datos maestro denominado «master dataset» (se puede ver como una gran lista de registros), así como del preprocesado de la información para la generación de vistas pre-computadas (Punto 1,2 de la Figura 2.7). Hadoop es un ejemplo.

Tiene dos funciones principales:

- Almacenar en HDFS el dataset maestro que es inmutable y constantemente crece.
- Crear vistas arbitrarias desde este dataset vía MapReduce. Esta computación se planifica y conforme llegan nuevos datos se agregan a las vistas en la siguiente iteración. Cada generación puede llevar horas.

2.8.2. Capa de servicio

La capa de servicio debe ser capaz de combinar los resultados del pre-procesado de datos en la capa de procesado batch con los datos expuestos a través la capa de velocidad. Además debe ser capaz de indexar y exponer los datos para que puedan ser consultados. (Punto 3 de la Figura 2.7)

2.8.3. Capa de velocidad

Debido al alto grado de latencia de la capa de procesado batch, y puesto que generalmente habrá ciertos datos que necesitarán estar expuestos de un modo más veloz, es necesaria una capa de procesado de información en tiempo real. A esta capa se le denomina capa de velocidad y es responsable del procesado en tiempo real del flujo de datos que necesita ser expuesto de manera inmediata. Conforme el flujo de datos está accesible la información es automáticamente analizada y pre-computada, generándose vistas que exponen estos datos para su consulta. (Punto 4 de la Figura 2.7)

Precisamente por la funcionalidad que pretende cubrir, los datos expuestos a través de la capa de velocidad son transitorios y sólo están disponibles de manera temporal; precisamente mientras no lo estén a través de la capa de procesado batch.

2.9. Características de Big Data

Cuando se trabaja con Big Data y se desea construir un sistema para representar una solución, hay que tener en cuenta varias características que se deberían cumplir [34]:

- Sistema de almacenamiento escalable que pueda capturar, manipular y analizar amplios volúmenes de datos.
- Una plataforma de computación, en algunos casos configurada especialmente para el análisis de datos a gran escala compuesta de varios nodos de procesamiento conectados a través de una gran velocidad.
- Un ambiente de gestión de datos donde se pueda cambiar su configuración desde un sistema de gestión de base de datos tradicional a un sistema de base de datos de paralelismo masivo con diversas distribuciones.
- Un ambiente de desarrollo de aplicaciones para facilitar el proceso de desarrollo, ejecución y verificación de códigos fuente para una aplicación. Este

ambiente debe incluir todas las facilidades para que se pueda implementar de manera correcta un sistema.

Al analizar una gran cantidad de datos, es necesario tener una infraestructura que soporte todas las funciones (adquirir, preparar, integrar, procesar) aplicables a los mismos. Algunas consideraciones en este sentido son:

- Escalabilidad: la posibilidad de crecer sin perder la calidad de su rendimiento
- Extensibilidad: que tan flexible es la arquitectura del sistema de almacenamiento para permitir que el sistema crezca sin la limitación de limites artificiales
- Accesibilidad: permite el acceso en simultáneo de diversos usuarios sin desmejorar el rendimiento del sistema.
- Tolerante a fallas: el ambiente de almacenamiento debe tener la capacidad de recuperarse mediante fallas
- Velocidad de E/S: se pueda satisfacer la demanda de lectura y escritura en simultáneo, cumpliendo con todas las peticiones.
- Integridad: el ambiente de almacenamiento sea fácil de integrar con otros ambientes como procesamiento, etc.

2.10. Principales usos

Son muchos los rubros en los que se puede aplicar el concepto y las tecnologías Big Data, entre ellos podemos destacar los siguientes [35]:

- Análisis de negocio (Business Intelligence): empresas están utilizando Big Data para mejorar sus procesos de negocios y potenciar el crecimiento de la empresa. El enfoque está dirigido a estudiar la relación cliente-empresa, mejorar la calificación de sus servicios y ayudar a la toma de decisiones. Gracias a todo esto las compañías pueden incrementar sus beneficios, optimizar costos e innovar con nuevos productos. [36]
- Business Intelligence (BI) se fundamenta en estructurar información empresarial útil y relevante para la toma de decisiones corporativas. Esta información suele ser conocida y focalizada en determinados ámbitos (ventas, producción, calidad, etc.), inclusive permite generar sistemas proactivos de previsión con datamining. En cualquier caso siempre basándose en datos estructurados y focalizados.

- Ambas técnicas son utilizadas por las empresas con un fin en común: obtener un valor agregado de la información que se dispone para ayudar a la toma de decisiones. La diferencia radica en que Big Data trata la información desde otra perspectiva, por la cantidad, complejidad y velocidad de la misma. Con BI se analizan datos conocidos para quien lo está usando, en cambio en Big Data se analiza todo tipo de datos sin que exista la necesidad de conocer su estructura. La idea que se está consolidando hoy en día, más allá de que los enfoques de una y otra técnica sean distintos, es que, en caso de que cada realidad particular lo amerite, BI y Big Data puedan coexistir, una como complemento de otra, con el objetivo de obtener los mayores réditos posibles para la empresa utilizando la información disponible.

- Análisis social: hoy en día con tantas redes sociales y acceso fácil a Internet es posible realizar encuestas y formularios para realizar estudios de investigación lo cual permite medir ciertos valores sociales de la población, por ejemplo: Trending Topic en Twitter, nos indica cuales son temas más hablados en Twitter.

- Uso médico y científico: el análisis de cantidades de datos considerables generados en streaming de eventos originados desde sensores, RFID o dispositivos médicos se puede implementar de manera sencilla, permitiendo el procesado de eventos complejos y la monitorización de varios pacientes en tiempo real.

- Estudios demográficos: estudio de la población desde un punto de vista científico, pudiendo identificar posibles causas de infecciones (buscando patrones con altas tasas de infección), redistribuir los recursos entre la población (por ejemplo, de 100.000 pacientes solo un 1 % tenía el 30 % de los recursos) o hacer predicciones de futuras pandemias.

- Seguridad: las nuevas tecnologías de Big Data han facilitado y mejorado en la seguridad. Cada vez hay más empresas que utilizan estas tecnologías para implementar soluciones referidas a la seguridad. Algunos ejemplos son la detección de ataques cibernéticos, localización de personas mediante información o entidades bancarias para detectar intentos de fraude. [37]

- Marketing para clientes: se analizan los perfiles de los clientes con el propósito de fórmulas campañas de marketing personalizado para influir en los comportamientos de compra de los clientes.

- Recomendación de sitios web: se analizan patrones en transacciones históricas combinadas con perfiles de los clientes para sugerirle temas adicionales como posibles compras.

2.11. Seguridad

En esta sección cuando hablamos de la seguridad en Big Data nos vamos a referir desde la perspectiva de proteger los datos que se tienen almacenados y procesados al momento de generar soluciones de Big Data. Se pueden tener dos cuestiones distintas, las que fijan la organización y la información de sus clientes en un contexto de datos amplios para preservar la privacidad que se tiene; y el uso de técnicas de Big Data para analizar e incluso predecir los incidentes de seguridad que se pueden filtrar. El uso a gran escala de infraestructuras en la nube, con una diversidad de plataformas de software y repartidas en extensas redes de computadoras aumenta la superficie de ataque de todo el sistema.

Mecanismos tradicionales de seguridad que se adaptan a asegurar datos estáticos a pequeña escala (en comparación con streaming) son insuficientes. Por ejemplo, los análisis para la detección de anomalías generarían demasiados valores atípicos. Del mismo modo, no está claro cómo se debe reforzar la procedencia en infraestructuras de la nube existentes, y el procesamiento en tiempo real requiere de tiempos de respuesta despreciables para asegurar la seguridad y privacidad.

El informe titulado «Top 10 Big Data Security and Privacy Challenges Report», publicado en noviembre de 2012, establecía 10 retos de seguridad que se deben tener en cuenta a la hora a afrontar proyectos Big Data [38]:

1. Asegurar la computación en marcos de programación distribuidos.
2. Implantar las mejores prácticas de seguridad en almacenamiento de datos no relacionales.
3. Asegurar el almacenamiento de datos y transacciones.
4. Validación en el punto final (End-point validation).
5. Seguridad en tiempo real y conformidad en la monitorización.
6. Data mining y analítica escalable y con capacidad de preservar la privacidad de los datos.
7. Control de acceso y seguridad de la comunicación con sistemas encriptados.

8. Control de acceso granular.

9. Auditorías granulares.

10. Gestión de la procedencia de los datos, en particular en los casos en los que es importante registrar el historial digital de los mismos.

Algunos de estos problemas que son específicos de los datos masivos surgen de la utilización de múltiples niveles de infraestructura (tanto en el almacenamiento y la computación) para el procesamiento de grandes volúmenes de datos. El uso de las nuevas infraestructuras de computación, tales como bases de datos NoSQL (para un buen rendimiento exigido por los volúmenes de datos grandes) que no han sido investigados a fondo por problemas de seguridad. La no escalabilidad de cifrado para grandes conjuntos de datos y de las técnicas de monitoreo en tiempo real. La heterogeneidad de los dispositivos que producen los datos, y la confusión con el exceso de diversas restricciones legales y de políticas que lleva a garantizar la seguridad y la privacidad.

El análisis de toda esta gran cantidad de información puede aportar una ventaja a las empresas para que puedan aprovecharla, pero a su vez tiene asociados algunos retos para la seguridad que deben tenerse en cuenta, la adopción de la tecnología que permita manejar Big Data debe ser pensada específicamente. Estructuras de cómputo distribuido, en los cuales intervienen múltiples plataformas y sistemas deben tener consideraciones especiales de seguridad, pues tanta diversidad puede dar a lugar a vulnerabilidades explotables por ciber-delincuentes.

Si bien el almacenamiento y el procesamiento en la nube es una alternativa demasiado interesante, es necesario contar con las garantías necesarias para que se mantenga la confidencialidad de la información. Una alternativa como la virtualización puede significar ventajas similares al procesamiento en la nube, pero con niveles de riesgo más controlables. Los sistemas más antiguos en las empresas suelen representar bastantes obstáculos para lograr la integración de sistemas, y muchas veces se opta por soluciones que simplifican los requerimientos de seguridad para lograr cumplir con las funcionalidades. Aunque son soluciones que facilitan la operación en el corto plazo, pueden convertirse en problemas más adelante. Como cualquier esquema de seguridad la educación de los empleados es necesaria para que se incorporen hábitos seguros en el manejo de la información, apoyados en soluciones que aseguren el acceso y manipulación de los datos.

Todos los cambios deben articularse desde las políticas de seguridad de la información, por lo cual se hace necesaria una revisión completa que permita identificar

aquello de debe ajustarse al nivel de riesgo aceptado por la empresa. Lograr analizar de forma oportuna los altos volúmenes de información para minimizar la ocurrencia de los riesgos en las empresas y maximizar las oportunidades propias del negocio es sin lugar a dudas el siguiente paso en el análisis de datos. Ya muchas empresas han dado ese paso, el reto está en implementarlo de forma segura.

2.12. Conclusiones

No solo la definición: «grandes cantidades de datos» (refiriendo al volumen de datos) significa Big Data sino que también la velocidad y la variedad forman parte de ese término. Esto conlleva a la problemática de no solo tener que analizar grandes volúmenes, sino también hacerlo de forma efectiva en cuanto al tiempo y contemplando la diversa variedad de datos.

La velocidad de recolección y procesamiento de datos es un factor que tiene gran impacto en soluciones de tiempo real, ya que debe responder de manera efectiva y rápida para mostrar resultados en el momento.

Las técnicas presentes en tecnologías Big Data surgieron desde mucho tiempo antes de la aparición de estas últimas, tales como procesamiento paralelo, Map Reduce, Grid Computing, entre otras. Entendemos que dado el gran volumen de datos que se maneja es necesario recurrir a técnicas de procesamiento paralelo y distribuido de modo de contrarrestar dicha característica.

La heterogeneidad de la información que se maneja es un aspecto que separa problemas de Big Data del resto. Esto genera una dificultad adicional al operador al momento de tratar los datos. Los sistemas tradicionales de bases de datos no están preparados para soportar la variedad de información que se quiere manipular, dando lugar a la aparición de nuevos sistemas que son definidos a partir de nuevos conceptos y principios. Surge la necesidad del uso de bases de datos no relacionales, permitiendo la manipulación de volúmenes espectaculares de datos que carecen de estructura, escapando al alcance de los sistemas tradicionales.

Pensamos que estamos frente a un quiebre en el tratamiento y uso de la información, que hasta hace un tiempo estaba enmarcado en un conjunto de técnicas y modalidades que pasan a ser insuficientes u obsoletas en algunos casos, para los nuevos desafíos. Dicho quiebre responde a la intensión por parte de los actores de obtener el mayor rédito posible de la información que se dispone.

Poder tratar la gran cantidad de información existente ha permitido a las empresas obtener una ventaja en cuanto a la toma de decisiones, no solo observando el comportamiento de los clientes sino mejorando sus productos. Los equipos directivos

de las empresas también emplean Big Data para controlar el rendimiento y hacer posible la toma de decisiones efectivas en tiempo real. Las ventajas del análisis de los grandes conjuntos de datos desestructurados no se restringen al sector privado, pueden mejorar también los servicios públicos.

Cabe destacar que las soluciones Big Data no sustituyen a las soluciones tradicionales de análisis de datos. Y de hecho en algunos casos es hasta contraproducente utilizar una solución Big data para un caso de uso que no la requiera.

En ciertas situaciones es factible combinar técnicas tradicionales con las de Big Data con el fin de maximizar el valor agregado que se pueda obtener de la información disponible. Un ejemplo de esto es la integración de Business Intelligence con Big Data.

Capítulo 3

Herramientas

En este capítulo se presentarán las principales herramientas para el desarrollo de soluciones de Big Data. En la sección 3.1 se definen los objetivos y tópicos a manejar en el capítulo. En la sección 3.2 se presenta el ecosistema Hadoop, el cual es un sistema que está enfocado en el procesamiento de datos a gran escala, alguna de sus herramientas que se ejecutan dentro del ecosistema se encontrarán en la sección 3.3. Las distribuciones que contienen un ambiente de desarrollo para algunas de las herramientas de Hadoop en la sección 3.4. En la sección 3.5 se presenta Apache Kafka, en la sección 3.6 Apache Storm y 3.7 Apache S4, las cuales son utilizadas para el procesamiento en tiempo real, luego una breve comparación de S4 vs Storm y por ultimo las conclusiones al capítulo.

3.1. Introducción

Dentro de los objetivos del proyecto se encuentra la investigación de herramientas para el procesamiento en tiempo real de grandes volúmenes de datos con la finalidad de conocer sus características, eficacia y utilidad para el caso de uso que deseamos presentar.

En este apartado se describen dos herramientas de uso extensivo para la toma de decisiones en tiempo real. Las mismas pertenecen al ecosistema Hadoop, y son denominadas Apache Storm y Apache Kafka.

Presentaremos algunas distribuciones que brindan un ambiente de desarrollo para una mejor y más amigable aplicación de los instrumentos proporcionados por Hadoop.

En el capítulo 4 se presenta un análisis de estas mismas herramientas, con el fin de determinar su aplicabilidad para la Empresa UnoWiFi.

3.2. Hadoop

Hadoop es un proyecto de Apache de código libre en el que colaboran un gran número de empresas (entre ellas Yahoo!, Facebook, Microsoft o Cloudera [39]). Su finalidad es englobar un conjunto de herramientas, aplicaciones y frameworks Java para el desarrollo de sistemas y utilidades de computación escalable y distribuida, permitiendo que las aplicaciones trabajen con miles de nodos y petabytes de información.

Es un sistema tolerante a fallas, se ejecuta sobre un hardware de bajo coste y maneja de forma automática la replicación de datos. Es importante destacar que Hadoop no es un sustituto de una base de datos. Almacena datos en archivos pero no los indexa. Si queremos encontrar algo, tenemos que ejecutar un trabajo MapReduce pasando por todos los datos.

La base de Hadoop se apoyó en los documentos originales de Google de MapReduce y File System, el sistema de ficheros distribuido de Google. Actualmente hay cientos de empresas usando Hadoop para sus procesos (tanto internos como de servicios al cliente). Entre las más destacadas encontramos a Microsoft, Facebook, Adobe, eBay, Google, IBM, Spotify o Twitter [40].

Hadoop fue creado por Doug Cutting y Mike Carafella en el año 2005 [41]. Ambos iniciaron en el año 2002 un proyecto de código libre para la búsqueda de términos en la web que llamaron «Nutch». El proyecto Nutch era prometedor, capaz de navegar e indexar millones de páginas. No obstante sólo funcionaba en un cluster de pocos nodos y constantemente tenía que haber una persona vigilando que no hubiera ningún error.

Tras unos meses trabajando en el proyecto Nutch, Google hizo público en octubre de 2003 un artículo llamado «Google File System» [42]. Mike Carafella se dio cuenta que el sistema que se proponía en el artículo era mejor que el sistema que habían montado en su proyecto. Ambos se pusieron a trabajar para construir un sistema parecido al descrito en el artículo de Google. Cuando ya tenían una primera versión disponible, Google hizo público en diciembre de 2004 otro artículo, «MapReduce» [43], y también les pareció buena idea para mejorar Nutch. En unos pocos meses, Cutting y Carafella habían construido en lenguaje Java el sistema de ficheros y el paradigma de procesamiento MapReduce que se convertirían en Hadoop. Migraron el proyecto Nutch al nuevo entorno y comprobaron una gran mejoría. Ahora Nutch podía trabajar con un cluster más grande de máquinas y no había graves problemas si alguna de las máquinas fallaba. Llamaron al nuevo entorno «Hadoop» y en poco tiempo se convirtió en un proyecto open source de Apache, que aumentó en gran

medida el número de colaboradores del proyecto [41].

Algunos ejemplos de herramientas compatibles con Hadoop son bases de datos NoSQL desarrolladas para trabajar con el sistema de ficheros HDFS, sistemas de monitorización del estado del cluster Hadoop, librerías con algoritmos de minería de datos que funcionan mediante el paradigma de programación MapReduce, herramientas que facilitan la población de datos del cluster Big Data, entre otras.

3.2.1. Arquitectura

El proyecto está formado por cuatro módulos:

- Hadoop Distributed FileSystem (HDFS): sistema de ficheros distribuido que provee acceso a los datos que necesitan las aplicaciones.
- Hadoop MapReduce: modelo de programación distribuido para procesar grandes volúmenes de datos.
- Hadoop YARN: gestor de recursos del cluster y planificador de ejecución de las aplicaciones.
- Hadoop Common: conjunto de utilidades que dan soporte a los otros tres módulos.

3.2.1.1. HDFS - Hadoop Distributed File System

HDFS está inspirado en el sistema de archivos de Google. El sistema de archivos distribuido de Hadoop tiene como propósito organizar los archivos en un espacio de nombres jerárquico. HDFS tiene dos diferencias fundamentales respecto otros sistemas de archivos, puede abarcar múltiples ordenadores y además se ejecuta en modo usuario. El sistema de archivos distribuido de Hadoop está diseñado para almacenar de forma fiable archivos de tamaño grande en clusters. HDFS almacena cada archivo como una secuencia de bloques. Todos los bloques del archivo excepto el último son del mismo tamaño. Los bloques pertenecientes a un archivo estarán replicados en diferentes clusters para la tolerancia a fallas. Tanto el tamaño de bloque como el factor de replicación se pueden configurar para cada archivo.

- NameNode: nodo único maestro, que controla el namespace del sistema de archivos y regula el acceso a los archivos de las peticiones clientes. Mantiene la trazabilidad de los bloques presentes en los DataNodes, esta información es utilizada para controlar la jerarquía de todo el sistema de archivo.

- DataNode: son los responsables de servir las lecturas y escrituras desde el sistema de archivos cliente, realiza las operaciones como abrir, cerrar y renombrar los archivos y directorios. Además realizan la creación de bloques, eliminación y replicación bajo las instrucciones del NameNode.

3.3. Herramientas del ecosistema Hadoop

A continuación se presentan un conjunto de herramientas de código abierto del ecosistema Hadoop, las cuales pueden ser usadas para crear soluciones de procesamiento de datos en tiempo real. El objetivo de esta sección no es explicar en profundidad todas las herramientas del proyecto Apache Hadoop sino más bien embarcarnos en un conjunto reducido (Ambari, Zookeeper, Chukwa, Flume, Cassandra, HBase, Pig, Mahount, Oozie, Kafka, Storm).

3.3.1. Utilidades

Se especificarán herramientas que permiten el uso práctico de los instrumentos de Hadoop.

3.3.1.1. Ambari

Ambari es una herramienta de Apache de código abierto, cuyo objetivo es simplificar el desarrollo de software para el suministro, gestión y seguimiento de los clusters Hadoop facilitando las tareas de administración mediante una interfaz, un conjunto de APIs y aplicaciones web [44].

Esta herramienta brinda a los administradores del sistema:

- Suministro de un cluster hadoop:
  - Ofrece un asistente paso a paso para la instalación de servicios Hadoop alrededor de cualquier cantidad de hosts.
  - Maneja la configuración de los servicios de Hadoop para el cluster.
- Administrar un cluster Hadoop:
  - Brinda una gestión central para iniciar, detener y reconfigurar los servicios de Hadoop en todo el cluster.
- Monitorear un cluster Hadoop:

- Brinda un panel de control para el seguimiento de la salud y estado del cluster Hadoop
- Aprovecha Ganglia y Nagios para la recolección de métricas, y para el sistema de alerta respectivamente.

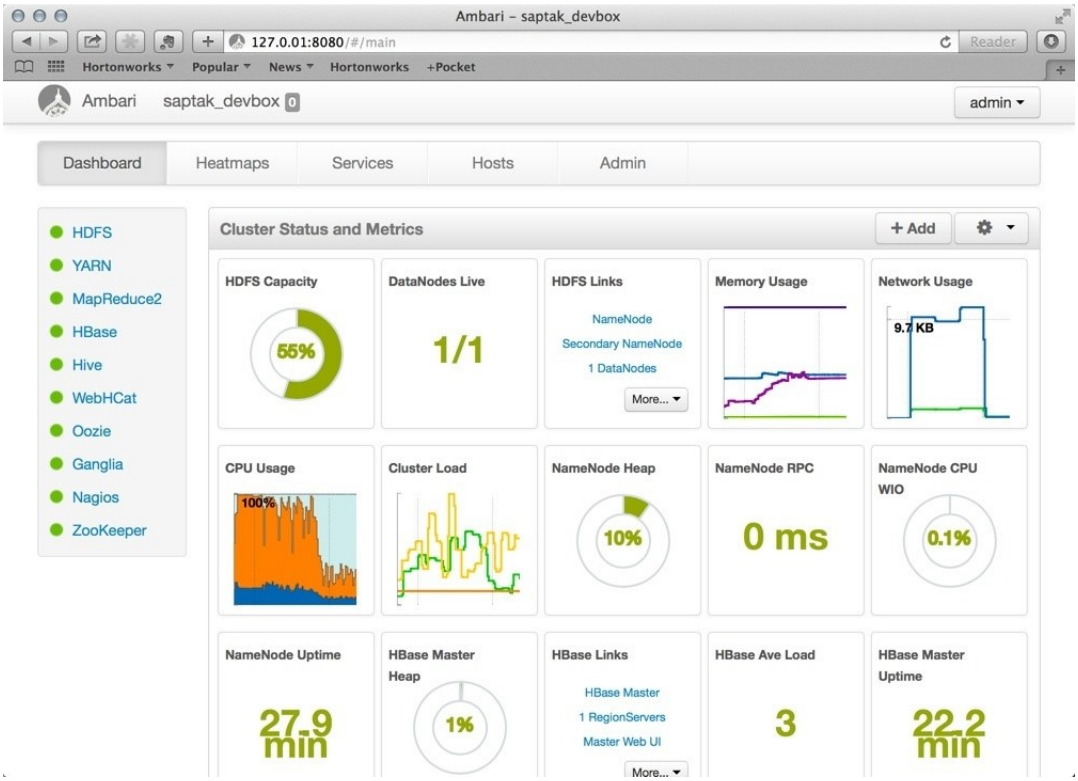


Figura 3.1: Consola de Apache Ambari en un navegador

3.3.1.2. Zookeper

Zookeeper es un servicio centralizado que se encarga de administrar y gestionar la coordinación entre procesos en sistemas distribuidos. El objetivo de Apache Zookeeper es el de facilitar a los desarrolladores la tarea de implementar funciones de mantenimiento entre sus procesos, como la sincronización entre estos, y ofrecer alta disponibilidad a través de servicios redundantes [45].

Funciona usando procesos distribuidos que se coordinan entre sí a través de un espacio de nombres jerárquico compartido que sigue el modelo de un sistema de archivos. Los datos se guardan en la memoria y se respaldan en un registro de confiabilidad. Zookeeper es muy rápido y puede manejar altas cargas que suelen verse en los protocolos de coordinación de tipo hablador utilizado por un gran número de procesos [46].

El espacio de nombres se compone de registros de datos (llamadas znodes, en la jerga Zookeeper) similares a los archivos y directorios. La principal diferencia entre un znode y un nodo de un sistema de ficheros es que los primeros pueden tener datos asociados, de manera que hasta los nodos directorios pueden contener información como si fueran ficheros normales y corrientes. Cada znode está limitado con el fin de evitar que Zookeeper sea manejado como un sistema de archivos. Esta necesidad parte de que los datos se mantienen en la memoria del sistema, por lo que se obtiene un rendimiento muy elevado a la hora de ejecutar las peticiones de los procesos cliente.

La implementación Zookeeper pone una prima en el alto rendimiento, la alta disponibilidad y acceso estrictamente ordenado. Los aspectos de rendimiento apuntan a que se puede utilizar en sistemas grandes y distribuidos. Los de alta disponibilidad que sea tolerante a fallas. El orden estricto se refiere a que sofisticadas primitivas de sincronización se pueden implementar en el cliente.

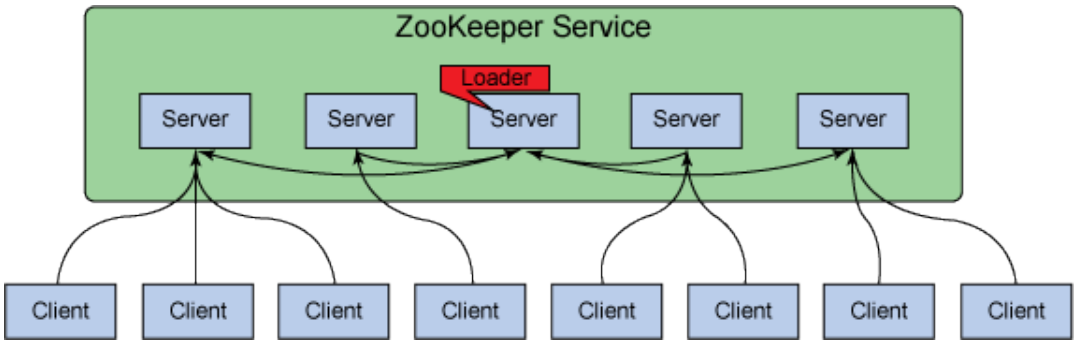


Figura 3.2: Servicio Zookeeper

Todos los servidores que componen el servicio Zookeeper deben saber el uno del otro. Mantienen una imagen de estado en memoria, junto con los registros de transacciones. Mientras que la mayoría de los servidores estén disponibles, el servicio estará disponible. Los clientes se conectan a un único servidor Zookeeper manteniendo una conexión TCP a través de la cual se envían peticiones, reciben respuestas y se visualizan eventos. Si la conexión TCP se cae, el cliente se conectará a un servidor diferente.

Mediante esta herramienta los desarrolladores de aplicaciones distribuidas tienen a su disposición una arquitectura y recursos para mantener sincronizados los servicios de un mismo cluster.

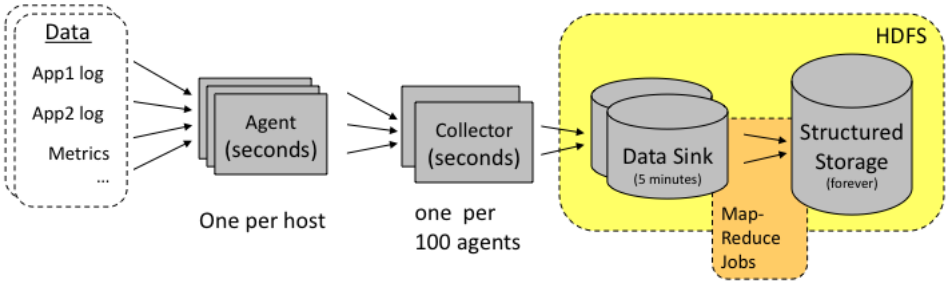


Figura 3.3: Arquitectura de Chukwa [47]

3.3.2. Recuperación de datos

Gracias al surgimiento de los instrumentos de Hadoop y de las bases de datos NoSql la necesidad de obtener grandes volúmenes de datos no estructurados ha permitido una significante evolución de instrumentos dirigidos a la recolección de datos.

3.3.2.1. Chukwa

Es un subproyecto Hadoop dedicado a la recopilación de registros a gran escala y el análisis de los mismos. Se construye en la parte superior del sistema de archivos distribuido Hadoop (HDFS) y el marco MapReduce, heredando su escalabilidad y robustez. También incluye un paquete de herramientas flexible y potente para mostrar el seguimiento y análisis de los resultados, con el fin de hacer el mejor uso de estos datos recogidos.

Hadoop está previsto para el trabajo óptimo sobre pocos ficheros de gran tamaño, no estando preparado para un tratamiento eficiente sobre sistemas de logs, compuestos por directorios con muchos ficheros pequeños dentro. Chukwa tiene como objetivo contrarrestar esta debilidad de Hadoop.

Su arquitectura consta de cuatro componentes [47].

- Agentes: procesos encargados de la captura de datos.
- Colectores: tienen como entradas los datos capturados por los agentes y los persisten de forma permanente.
- Tareas MapReduce para trabajar con la información.
- HICC: interfaz web que permite la visualización de los datos.

Agentes y Adaptadores Los agentes son los encargados de capturar los datos y enviárselos a los colectores. Los adaptadores son fuentes de datos que pueden ser configuradas dinámicamente a partir de módulos dentro de los agentes, permitiendo así que la variedad de los datos no genere un problema considerable en esta tarea.

Los datos emitidos por los adaptadores se envían por paquetes llamados Chunks. Estos paquetes están formados por una secuencia de bytes (los datos a enviar) y metadatos (que describen los datos). Los metadatos suelen estar generados por el agente pero hay un par de ellos que sí es necesario que los defina el usuario: el nombre del cluster y los tipos de los datos que se están enviando.

Colectores Después que se genera un chunk se lo envía a los colectores vía HTTP. Estos son los encargados de escribir en HDFS los chunks recibidos. La escritura se realiza en un fichero con la secuencia de datos serializados, llamado fichero sink. Una vez realizada esta tarea se los marca como disponibles para ser procesados y se comienza la escritura de otro fichero desde el comienzo.

3.3.2.2. Flume

Flume es un servicio distribuido para la recolección, agregación y desplazamiento de grandes volúmenes de datos hacia un almacenamiento centralizado. Ofrece una arquitectura basada en la transmisión de datos por streaming altamente flexible y configurable pero a la vez simple. Al tener un origen de datos configurable, se adapta prácticamente a cualquier tipo de situación: monitorización de logs, descarga de información de redes sociales o mensajes de correo electrónico, entre muchas otras. Los destinos de los datos también son altamente configurables de manera que el uso de Flume no va ligado exclusivamente con HDFS o incluso Hadoop. La arquitectura de Flume [48] está basada en agentes, que son procesos encargados de recolectar datos y enviarlos a su destino. A su vez, el flujo de datos viene determinado por una unidad llamada evento que, como pasaba con los chunks de Chukwa, está formado por datos y metadatos.

Un agente tiene tres componentes:

- Source: es el encargado de recibir los datos, convertirlos en eventos y escribirlos en el channel. Hay varios tipos de sources.
- Channel: es un almacenamiento pasivo que recibe los eventos del source y espera a que el sink lo consuma, es decir que los sources añaden eventos mientras que los sinks los retiran. Dependiendo del tipo de tolerancia a fallas y del rendimiento que se desea hay distintos tipos de channels.

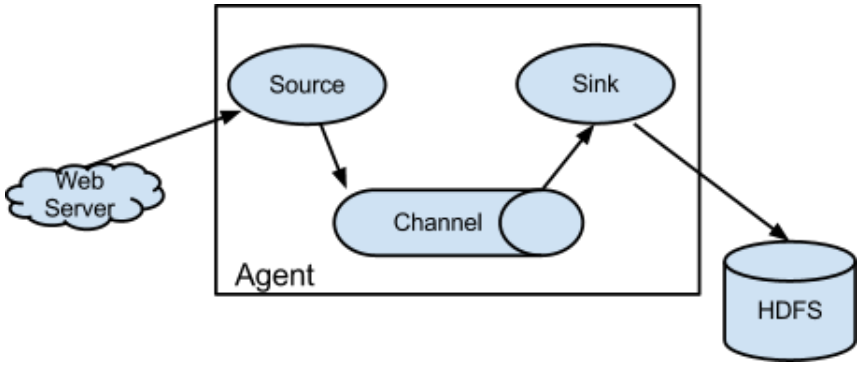


Figura 3.4: Arquitectura de Apache Flume [48]

- Sink: se encarga de leer los eventos del channel y dependiendo de su tipo enviarlo a diferentes sitios.

3.3.3. Almacenamiento de datos

Aunque Hadoop ya cuenta con el sistema de almacenamiento HDFS, existen varios proyectos que complementan, e incluso trabajan conjuntamente con el sistema de ficheros para aumentar las características y la flexibilidad de Hadoop entorno a las necesidades de sus usuarios.

3.3.3.1. Cassandra

Cassandra es una base de datos NoSQL, mayormente desarrollada por Datastax aunque empezó como un proyecto de Facebook, escrita en Java y open-source (también es un proyecto Apache). Entre sus características se encuentra la de ofrecer alta disponibilidad de los datos junto con una gran capacidad para escalar linealmente, además de ser tolerante a fallos (no tiene un punto de fallo único) y compatible con hardware de bajo presupuesto o infraestructuras en la nube. En cuanto al modelo de datos Cassandra es una base de datos distribuida orientada a columnas (inspirada en BigTable de Google y en Dynamo de Amazon), pensada especialmente para datos semi-estructurados (aunque también se puede adaptar a los estructurados y no estructurados) y que contiene varios espacios de claves con el siguiente esquema:

- Los espacios de claves están formados por un número determinado de familias de columnas -también conocidas como supercolumnas.

- Cada familia de columnas está identificada con una clave y tiene por valor un grupo indeterminado de filas.

- Cada fila, también identificada por una clave, tiene por valor distintas columnas (que pueden añadirse dinámicamente y por lo tanto no hay un número definido de columnas). Las columnas de cada fila son independientes de las que haya en las otras filas.

- Cada columna es una tupla (nombre, valor, stamp). Cada vez que se modifica un valor también se cambia el timestamp, por lo que se puede hacer un seguimiento de los cambios de un valor dado.

3.3.3.2. HBase

Al igual que Cassandra, HBase es una base de datos NoSQL de código abierto Apache basada en Hadoop escrita en Java que proporciona un acceso aleatorio y una coherencia fuerte para grandes cantidades de datos no estructurados y semiestructurados [49]. Se modeló en BigTable de Google [50] y es una base de datos orientada a las familias de columnas.

Los datos se almacenan en las filas de una tabla, mientras que los datos de una fila se agrupan por familia de columnas. HBase es una base de datos sin esquemas en el sentido de que no es preciso definir las columnas o el tipo de datos almacenados en ellas se definan antes de usarlos.

Mike Cafarella lanzó por primera vez el código fuente abierto en 2007 [51].

Escala linealmente para gestionar petabytes de datos en miles de nodos. Puede basarse en la redundancia de datos, el procesamiento de lotes y otras características proporcionadas por aplicaciones distribuidas en el ecosistema Hadoop.

3.3.3.3. Comparación entre HBase y Cassandra

Característica	HBase	Cassandra
Tipo	Almacenamiento orientado a columnas	Almacenamiento orientado a columnas
Origen	BigTable. HBase se construye en la parte superior del ecosistema de Hadoop	BigTable y DynamoDB
Particionamiento	Fuerte consistencia y particionamiento ordenado	Disponibilidad y particionamiento ordenado y aleatorio

Tabla 3.1: Cuadro comparativo HBase - Cassandra, parte 1

Escalabilidad	Lineal	Lineal
Estrategia de replica	Regiones de servidores	Estrategia simple con un datacenter
Detención por error automática	Si	Si
Acceso a los datos	Java y Restful APi	CQL (similar a SQL). Drivers en C# y Java
Operaciones optimizadas	Lectura	Escritura
Cache	Si	Si
Procedimientos de servidor	Si, triggers y store procedures	No
Compañías	Facebook, Twitter, etc.	Netflix, Ebay, Twitter

Figura 3.5: Cuadro comparativo HBase - Cassandra, parte 2

3.3.4. Tratamiento de datos

Hadoop también dispone de un buen número de herramientas para tratar los datos, las que serán presentadas a continuación.

3.3.4.1. Pig

Apache Pig es una plataforma dirigida al análisis de grandes volúmenes de datos. El análisis se realiza a partir del desarrollo de procedimientos de alto nivel sobre datos semiestructurados. El lenguaje utilizado se denomina PigLatin, el cual permite realizar consultas similares a las que se pueden hacer con SQL [52].

Esta herramienta consta de un compilador que se encarga de traducir las sentencias PigLatin en tareas MapReduce sin la necesidad de que el programador deba programarlas por separado. Además, se encarga de optimizar automáticamente los programas hechos por los usuarios, respetando la coherencia de los datos.

Presenta dos modos de trabajo:

- Modo Batch: se le pasa por parámetro al comando «pig» un fichero script que contiene el flujo de datos.

- Modo Interactivo: se trabaja en una terminal de línea de comando enviando sentencias una a una de forma interactiva

3.3.4.2. Mahout

Apache Mahout es un proyecto para crear aprendizaje automático y data mining usando Hadoop. Permite descubrir patrones en grandes conjuntos de datos. Tiene algoritmos de recomendación, clustering y clasificación. [53]

En la actualidad esta herramienta da soporte a los siguientes problemas:

- Minería de recomendaciones: identifica y aprende los gustos de los usuarios a través de su comportamiento.
- Clustering: dado un conjunto de documentos se encarga de agruparlos, diferenciarlos y clasificarlos.
- Minería de frecuencia de ítems: a partir de un grupo de ítems identifica cuales son los más comunes en cuanto a la frecuencia de aparición.

3.3.4.3. Oozie

Oozie es un planificador de flujos de trabajo para sistemas que realizan trabajos o procesos Hadoop. Su uso no exige un conocimiento técnico experto del tema por parte de los usuarios ya que presenta una interfaz de alto nivel que gracias a su abstracción permite a estos usuarios realizar flujos de trabajo complejos. [54]

Funciona como un motor de flujos de trabajo a modo de servicio que permite lanzar, parar, suspender, retomar y volver a ejecutar una serie de trabajos Hadoop (no solamente MapReduce) basándose en ciertos criterios, ya sean temporales o de disponibilidad de datos. Los flujos de trabajo Oozie son grafos acíclicos dirigidos (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.

Los flujos tienen dos tipos de nodos:

- Nodos de control: definen el inicio y el final del flujo (start, fail, end) así como mecanismos para el flujo de ejecución (decisión, unión, división)
- Nodos de acción: ejecutan alguna tarea de procesamiento, ya sea un trabajo MapReduce, una acción HDFS, instrucciones PIG o incluso otro grafo Oozie.

3.4. Distribuciones del ecosistema Hadoop

Se puede configurar el ecosistema Hadoop en un sistema de manera manual. Hoy en día existen varias organizaciones que se dedicaron a brindar un ambiente preparado

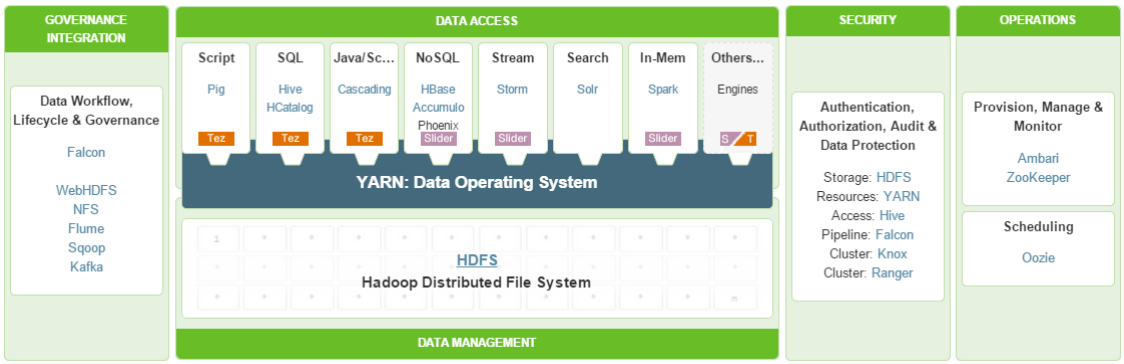


Figura 3.6: Herramientas pertenecientes a la distribución Hortonworks [55]

con Hadoop para su uso. En este apartado se analizan algunas de las distribuciones Hadoop más populares, como lo son Hortonworks y Cloudera.

3.4.1. Hortonworks

Hortonworks Data Platform (HDP) es una distribución de Hadoop completamente de código abierto. Actualmente existen varias versiones, la más reciente HDP 2.2. La nomenclatura de las versiones está estrechamente relacionada con Hadoop.

Hortonworks básicamente es una imagen de un sistema Linux más precisamente CentOS que contiene todas las herramientas pertenecientes al ecosistema Hadoop, lo que facilita el uso de los mismos. Esta plataforma esta apta para Windows, Linux y MAC.

En la figura 3.7 se muestra como fueron evolucionando las herramientas en las distintas versiones de Hortonworks y qué herramientas se han sido agregando con el transcurso del tiempo.

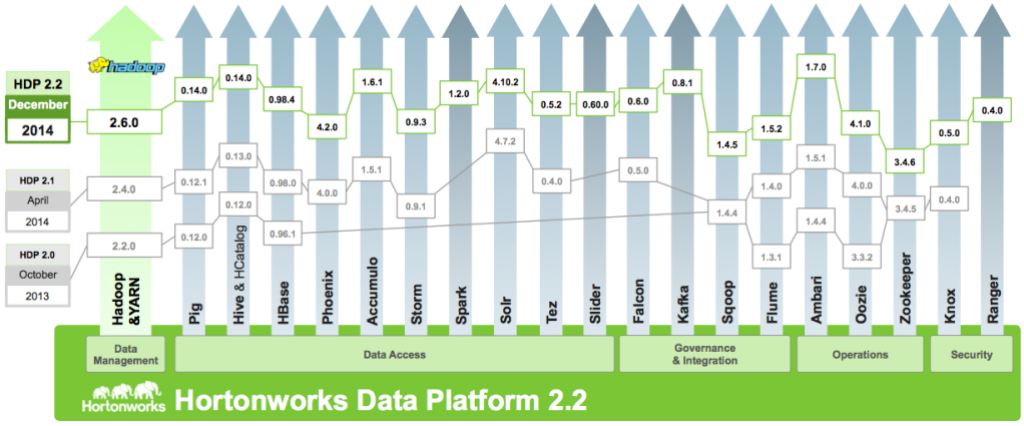


Figura 3.7: Evolución de las versiones de las herramientas del ecosistema Hadoop en la distribución de Hortonworks [55]

3.4.2. Cloudera

Cloudera proporciona una plataforma flexible, escalable e integrada que facilita la gestión del alto volumen y variedad de los datos. Permite desplegar y gestionar Hadoop y proyectos relacionados, manipular y analizar sus datos manteniéndolos seguros y protegidos. Cloudera ofrece los siguientes productos y herramientas [56]:

- CDH: la distribución de Cloudera Hadoop y otros proyectos de código abierto relacionados, incluyendo Impala y Cloudera Search. CDH también proporciona seguridad y la integración con numerosas soluciones de hardware y software.
- Cloudera Manager: una sofisticada aplicación utilizada para implementar, administrar, supervisar y diagnosticar problemas con sus despliegues CDH.
- Cloudera Navigator: una herramienta de gestión de datos de extremo a extremo para la plataforma CDH. Permite a los administradores, gerentes y analistas de datos, para explorar las grandes cantidades de datos en Hadoop. La auditoría robusta, gestión de datos y la gestión del ciclo de vida en Cloudera Navigator permiten a las empresas a que se adhieran al cumplimiento estricto y los requisitos reglamentarios.
- Cloudera Impala: Un motor de procesamiento masivamente paralelo de SQL para análisis interactivos e inteligencia de negocios. Su arquitectura altamente optimizada hace que sea ideal para las consultas tradicionales de estilo BI uniones, agregaciones, y subconsultas.

3.4.2.1. CDH

Esta distribución proporciona los elementos básicos de Hadoop (almacenamiento escalable y computación distribuida) junto con una interfaz de usuario basada en Web y capacidades empresariales vitales. CDH es de Apache, de código abierto. Proporciona [57]:

- Flexibilidad: almacenar cualquier tipo de datos y manipular con una variedad de diferentes marcos de computación, incluyendo el procesamiento por lotes, SQL interactivo, búsqueda de texto libre, aprendizaje automático y cómputo estadístico.
- Integración: se puede poner en marcha rápidamente en una plataforma Hadoop que trabaja con una amplia gama de soluciones de hardware y software.
- Seguridad: procesar y controlar datos sensibles.

- Escalabilidad: habilitar, escalar y ampliar una alta gama de aplicaciones para satisfacer las necesidades.
- Alta disponibilidad: realizar tareas críticas de negocio con confianza.
- Compatibilidad: se aprovecha al máximo la infraestructura de TI existente.

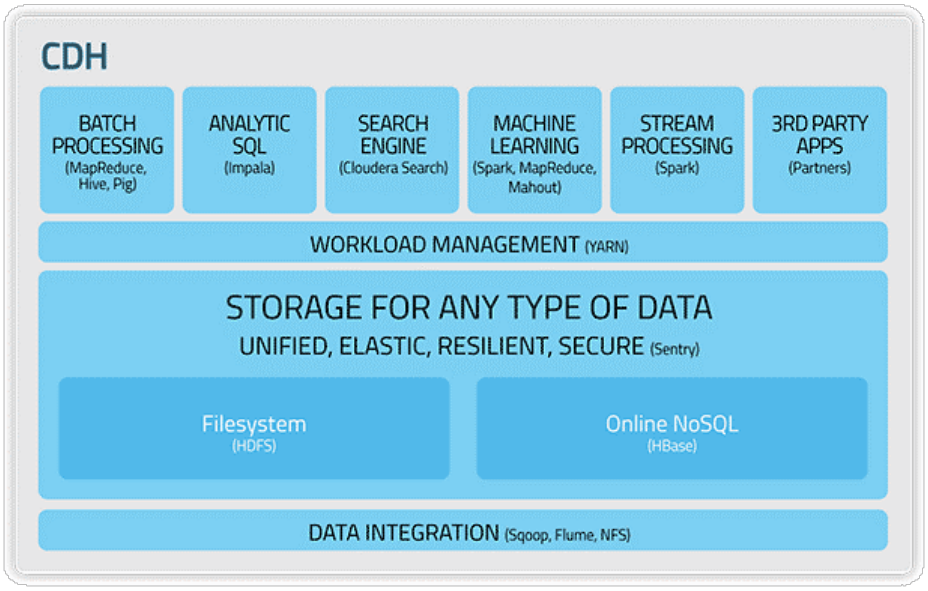


Figura 3.8: Arquitectura de Cloudera [58]

3.4.2.2. Cloudera Manager

Cloudera Manager es una aplicación web que facilita la administración de todo el cluster Hadoop, brindando un asistente de instalación automatizada de Cloudera CDH, y permitiendo la administración de todos los servicios instalados en el cluster así como la monitorización a tiempo real de dichos servicios. También se permite la monitorización del estado de salud del cluster, para que de forma rápida y visual se pueda detectar cualquier error en alguno de los nodos del cluster y solucionarlo de la forma más rápida posible.

3.4.2.3. Cloudera Navigator

Cloudera Navigator es una extensión de Cloudera Manager, disponible únicamente bajo suscripción, que facilita las gestiones de administración sobre los datos almacenados en el cluster Hadoop. Cloudera Navigator permite verificar el acceso de los usuarios y grupos de usuario a los ficheros y directorios almacenados en HDFS, permitiendo únicamente el acceso a quienes tengan credenciales y permisos suficientes. Además facilita la auditoría de datos al mantener un registro completo de todos

los accesos realizados a los datos almacenados en HDFS, Hive y HBase, pudiendo comprobar que usuarios han accedido a que datos y cuando se ha realizado dicho acceso.

3.4.2.4. Cloudera Impala

Es una base de datos relacional MPP (Massively Parallel Porcessing) que permite ejecutar consultas SQL interactivas directamente sobre los datos almacenados en Hadoop. Impala es un motor de consulta SQL completamente escalable que permite hacer consultas sobre datos almacenados en HDFS y en la base de datos NoSQL HBase. Facilita el acceso a Hadoop y a todas las herramientas de Business Intelligence al permitir realizar consultas SQL mediante conectores JDBC/ODBC en tiempo casi real.

3.4.3. Comparativa Cloudera - Hortonworks

Performance y escalabilidad

	Hortonworks	**Cloudera**
Ingestión de datos	Batch	Batch
Arquitectura de metadata	Centralizada	Centralizada
HBase - Performance	Picos de latencia	Picos de latencia
Aplicación No-SQL	Principalmente aplicaciones batch	Principalmente aplicaciones batch

Confianza

	Hortonworks	**Cloudera**
Alta disponibilidad	Recuperación de fallo único	Recuperación de fallo único
MapReduce Alta disponibilidad	Reinicia trabajos	Reinicia trabajos
Actualización	Tiempo de inactividad planeado	Tiempo de inactividad planeado
Replicación	Datos	Datos
Recuperación de desastres	No	Copia de archivo programada

Gestión

	Hortonworks	**Cloudera**
Herramientas de gestión	Ambari	Cloudera Manager
Soporte de volumen	No	No
Alarmas, alertas	Si	Si
Integración con REST API	Si	Si

Acceso a datos

	Hortonworks	**Cloudera**
Acceso a sistema de archivos	HDFS, NFS solo lectura	HDFS, NFS solo lectura
Archivos de Entrada/Salida	Solo anexar	Solo anexar

3.5. Apache Kafka

En esta sección introduciremos a la herramienta Apache Kafka, un sistema de almacenamiento. Explicaremos como está compuesta y su funcionamiento.

3.5.1. Introducción

Apache Kafka es un sistema de almacenamiento de mensajes publicador/suscriptor (o productor/consumidor) distribuido, particionado y replicado, desarrollado originalmente por LinkedIn Corporation. Sus características lo vuelven muy rápido en lecturas y escrituras convirtiéndolo en una herramienta excelente para comunicar streams de información. Los streams son generados a gran velocidad, en grandes volúmenes y deben ser gestionados por una o varias aplicaciones [59].

Se destacan las siguientes características:

- Funciona como un servicio de mensajería, categoriza los mensajes en topics.
- Los procesos que publican se denominan «producers» (productores) y los suscriptores son los «consumers» (consumidores) de los topics.

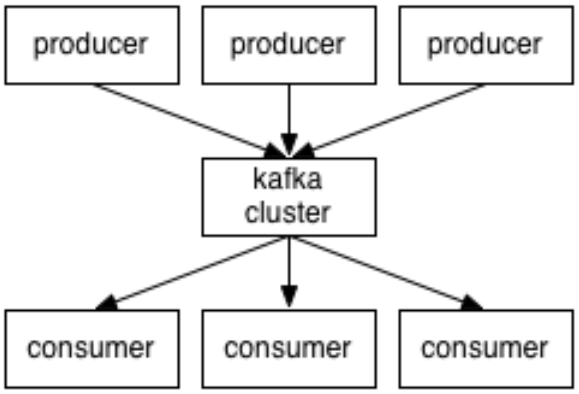


Figura 3.9: Diagrama de productores/consumidores de Apache Kafka

- Kafka se ejecuta en un cluster de uno o varios servidores llamados «brokers».
- 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 como Java, Scala, Python, Ruby, C++.
- Escalable y tolerante a fallas.
- Se puede utilizar para servicios de mensajería (tipo ActiveMQ o RabbitMQ), procesamiento de streams, web tracking, trazas operacionales, entre otras.
- Escrito en Scala.

En alto nivel, los productores envían mensajes a través de la red al cluster, donde a su vez estos se los reenvían a los consumidores.

3.5.2. Componentes

En esta subsección explicaremos en alto nivel los componentes que forman la arquitectura de Apache Kafka.

3.5.2.1. Topics

Un «topic» o una categoría en su traducción al español, representa al nombre de una categoría o grupo en la que se publican mensajes. Apache Kafka escribe en un «topic» como si fuera un RAID de discos duros que se denominan particiones. Cada

partición es una secuencia, ordenada e inmutable de mensajes que están continuamente apilados. Los mensajes en las particiones tienen asignados un identificador que es denominado «offset», este identifica de forma única al mensaje en su partición. Son estas particiones las que permiten paralelizar la escritura y gestionar la redundancia, la tolerancia a fallas del sistema y el balanceo de carga.

Las particiones del registro se distribuyen a través de los servidores del cluster Kafka con el manejo de datos como las solicitudes de una parte de las particiones de cada servidor. Cada partición se replica a través de un número configurable de servidores para la tolerancia a fallos [59].

Para cada «topic», el cluster Kafka mantiene un registro con particiones que tiene el aspecto ilustrado en la Figura 3.10 :

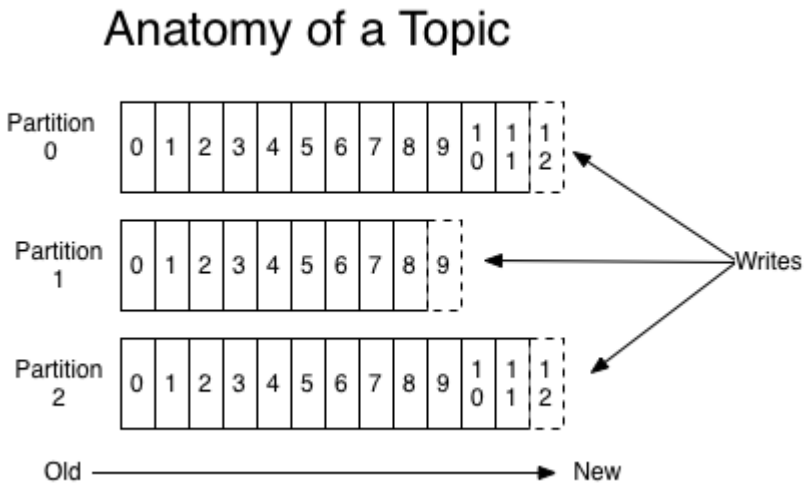


Figura 3.10: Diagrama de anatomía de un topic [60]

Apache Kafka no basa sus mensajes en JMS (Java Message Service), sino que usa un protocolo binario sobre TCP con su propia especificación. Un mensaje no es más que un byte payload y un checksum para comprobar que no se hayan corrompido. Gracias a esto, se limita el tráfico de datos por la red y la latencia de los mensajes. Lo más llamativo de Apache Kafka es cómo se realiza la lectura y borrado de los mensajes:

- La lectura de los mensajes por parte de los consumidores se basa en un offset desde el inicio para leer el mensaje correspondiente en cada momento. Esto implica que una lectura no tiene como consecuencia el borrado del mensaje leído por lo que se mejora el rendimiento de manera considerable.

- Por otro lado, el borrado de los mensajes se realiza mediante una especie de proceso desatendido que se encarga de hacer un tipo de limpieza de base de datos en los ficheros de logs (particiones) en el tiempo que se configure. Esto es posible ya que la estructuración de las particiones se fundamenta en la jerarquía de ficheros con indirección como en los sistemas operativos Linux cuando los ficheros eran muy grandes.
- El cluster Kafka mantiene los mensajes publicados, tanto si son consumidos como si no, por un período de tiempo configurable. Después, de este tiempo los mensajes son borrados y se libera la memoria que estaba reservada.

3.5.2.2. Productores

Los productores publican mensajes a un determinado topic que representa el tema y es responsable de elegir qué mensaje asignar a la partición en el topic. Esto se puede hacer mediante round-robin simplemente para equilibrar la carga o puede hacerse de acuerdo con alguna función de partición semántica (por ejemplo sobre la base de alguna clave en el mensaje).

3.5.2.3. Consumidores

Los consumidores de Kafka pueden tener instancias en distintos procesos o máquinas. Cada consumidor tiene un identificador de grupo.

Existen dos modelos: de cola y publicación-suscripción.

- En un sistema de colas, cada grupo de consumidores puede leer desde un servidor y obtener esos mensajes.
- En un sistema de publicación-suscripción el mensaje se difunde a todos los consumidores.

Normalmente, cada grupo tiene múltiples instancias de consumidor para proporcionar escalabilidad y alta disponibilidad. Los consumidores usan Zookeeper para llevar un conteo del offset por los que van leyendo y cuál es el offset actual de un topic.

3.5.3. Arquitectura

La figura 3.11 presenta los componentes y cómo están comunicados entre sí.

Podemos observar un cluster Kafka que contiene tres brokers. En este cluster hay un topics (Topic-1) y tres particiones para ese topic. También, encontramos tres

consumidores, cada uno consumiendo de un grupo diferente, y encontramos a dos productores publicando en todas las particiones. Tanto los consumidores, como los productores están operando en el Topic-1 pero en diferente partición. Podemos observar en la parte derecha de la figura cómo cada consumidor lee de una partición. Una cualidad de los brokers es que no son responsables del seguimiento de los mensajes contribuyendo la escalabilidad y el rendimiento, la responsabilidad es del consumidor.

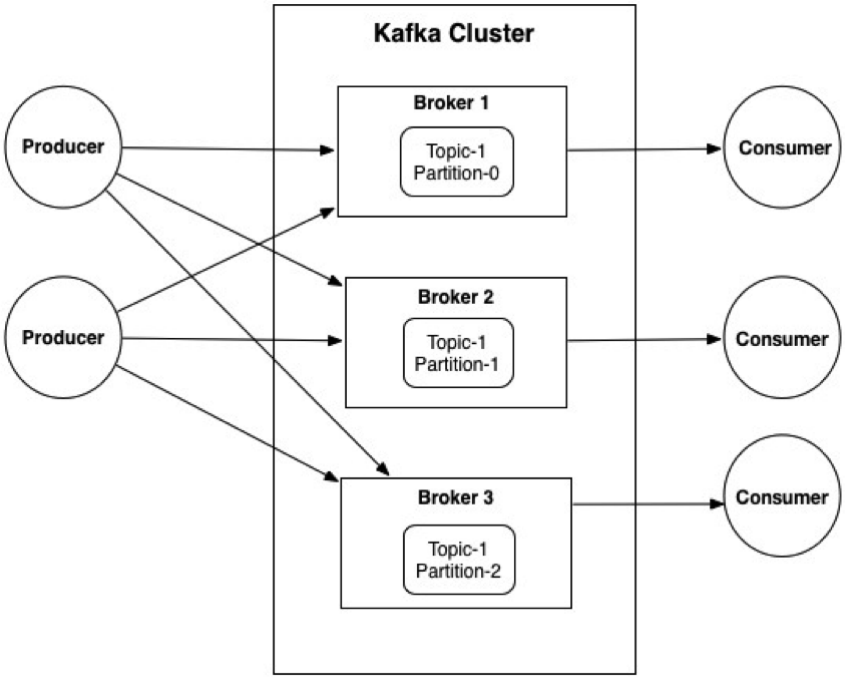


Figura 3.11: Arquitectura de Apache Kafka

3.5.4. Funcionamiento

En esta subsección se presenta el funcionamiento de Apache Kafka, al ser un sistema de almacenamiento de grandes volúmenes se desea comprender su modo de gestión.

3.5.4.1. Almacenamiento

Cada partición de un topic corresponde a un registro lógico. Físicamente, un registro se implementa como un conjunto de archivos de segmentos de igual tamaño. Cada vez que un productor publica un mensaje a una partición, el broker simplemente añade el mensaje en el archivo en el último segmento. Los segmentos de los archivos se escriben en disco después de un número configurable de mensajes hayan sido publicados o después de una cierta cantidad de tiempo transcurrido. Los mensajes son expuestos a los consumidores después de que se elimina. A diferencia del sistema

tradicional de mensajes, un mensaje almacenado en el sistema de Kafka no tiene identificadores de mensajes.

Los mensajes son expuestos por el desplazamiento en el registro lógico. Esto evita la sobrecarga proveniente de mantener grandes estructuras de índice auxiliares utilizadas para asignar identificadores a los mensajes. Los identificadores de mensaje son incrementales, pero no consecutivos. Para calcular el ID del mensaje siguiente añade la longitud del mensaje actual a su lógica de desplazamiento.

Los consumidores siempre consumen mensajes de una forma secuencial en una partición particular y si el consumidor reconoce el desplazamiento de un mensaje en particular implica que el consumidor ha consumido todos los mensajes anteriores. Los consumidores realizan una solicitud de extracción asíncrona al «broker» para obtener un buffer de bytes listo para consumir. Cada solicitud de extracción asincrónica contiene el desplazamiento del mensaje a consumir.

3.5.4.2. Distribución

En esta parte entra en juego Apache Zookeeper cuyo funcionamiento fue descrito en el apartado 3.3.1.2. Kafka lo utiliza debido a su capacidad coordinación. Cada broker está coordinado con otros broker mediante Zookeeper. Productor y consumidor son notificados mediante Zookeeper sobre la presencia del nuevo broker en el sistema o el fracaso de uno. De acuerdo con la notificación recibida, el productor y el consumidor toman la decisión y empieza a coordinar su trabajo con algún otro broker. En general, la arquitectura del sistema Kafka se muestra a continuación en la Figura 3.12.

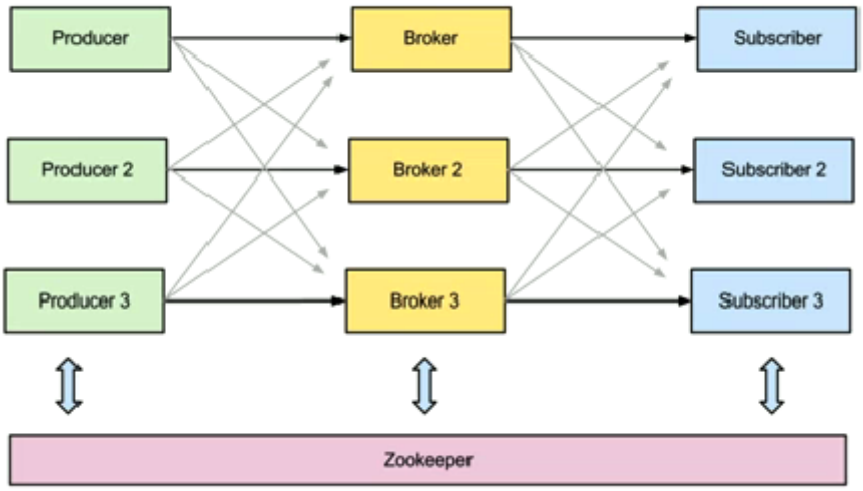


Figura 3.12: Distribución de Kafka con Zookeeper

3.6. Apache Storm

En esta sección explicaremos cómo funciona la herramienta Apache Storm, la cual es una herramienta para el procesamiento de datos en tiempo real.

3.6.0.3. Introducción

Esta tecnología fue impulsada por Nathan Marz, quien la implantó de forma exitosa en Twitter, fue liberada en 2011 y adquirida por Apache a finales del 2013 donde se continúa su desarrollo [61].

Uno de los más interesantes ejemplos de su uso es la generación de información de tendencias. Twitter extrae tendencias emergentes a partir de los tweets publicados y las mantiene a nivel local y nacional. Esto significa que a medida que una historia comienza a surgir, el algoritmo de temas de tendencia de Twitter identifica el tema en tiempo real. Este algoritmo se implementa en Storm como un análisis continuo de datos de Twitter.

Lo que diferencia a Storm de otras soluciones de Big Data es su paradigma. Hadoop es fundamentalmente un sistema de procesamiento por lotes. Los datos se introducen en el sistema de archivos de Hadoop (HDFS) y se distribuyen a través de los nodos para su procesamiento. Cuando el procesamiento está completo, los datos resultantes regresan a HDFS para ser utilizados por el originador. Storm soporta la construcción de topologías que transforman secuencias de datos sin finalizar. Esas transformaciones, a diferencia de los trabajos de Hadoop nunca se detienen, sino que continúan procesando datos conforme van llegando [62].

3.6.0.4. Componentes

Apache Storm cuenta con varios componentes. En este apartado explicaremos cada uno de ellos y cómo interactúan entre sí para brindar un correcto funcionamiento.

Entre los conceptos más importantes a considerar para el manejo de esta herramienta se destacan los siguientes:

- Tuplas: una lista ordenada de elementos.
- Streams: flujo de datos. Es una secuencia ilimitada de tuplas. Storm proporciona las primitivas para la transformación de un stream a otro stream nuevo.
- Spouts: origen de un stream.
- Bolts: reciben, procesan y emiten streams.

- Topologías: para realizar un procesamiento en Storm se necesita construir topologías, estas son un grafo computacional, cada nodo representa al procesamiento lógico, cada vinculo representa como es enviada esos resultados de procesamiento lógico. En la figura 3.13 podemos ver cómo está relacionado con los demás componentes.

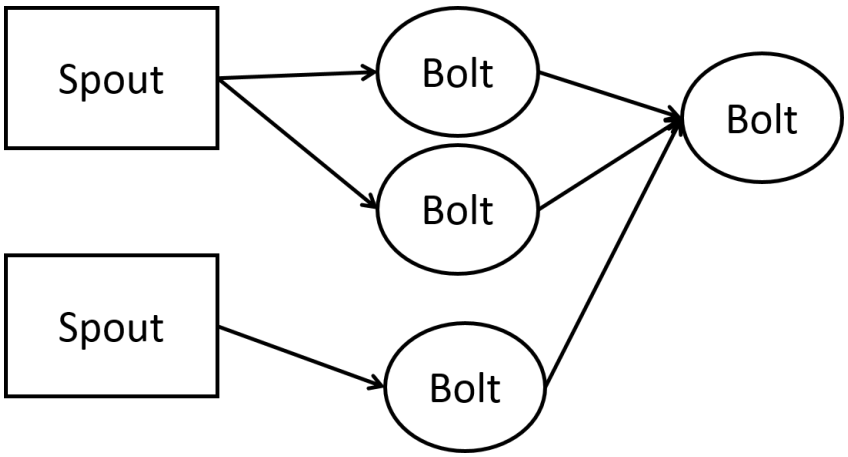


Figura 3.13: Topología de Storm

Un cluster Storm está compuesto por los siguientes elementos [63]:

- Nodo Nimbus: el nodo principal, es similar al «Jobtracker» de Hadoop. Nimbus es responsable de la distribución de código en todo el cluster, la asignación de tareas a las máquinas y el seguimiento de los fracasos.
- Nodos supervisor: inicia / detiene trabajadores y se comunica con Nimbus través Zookeeper (proyecto de Apache que proporciona una infraestructura centralizada y de servicios que permiten la sincronización)
- Nodos Zookeeper: coordina al cluster.

Otra funcionalidad que brinda Storm es una interfaz de usuario accesible desde un navegador llamada Storm UI, la cual proporciona estadísticas e información acerca del cluster y las topologías que ejecutan en él.

Storm implementa un conjunto de características que lo definen en términos de rendimiento y confiabilidad. Utiliza ZeroMQ para el pasaje de mensajes, lo cual elimina las colas intermedias y permite que los mensajes fluyan directamente entre las tareas mismas. Detrás de la mensajería se encuentra un mecanismo automatizado eficaz para la serialización de tipos primitivos de Storm.

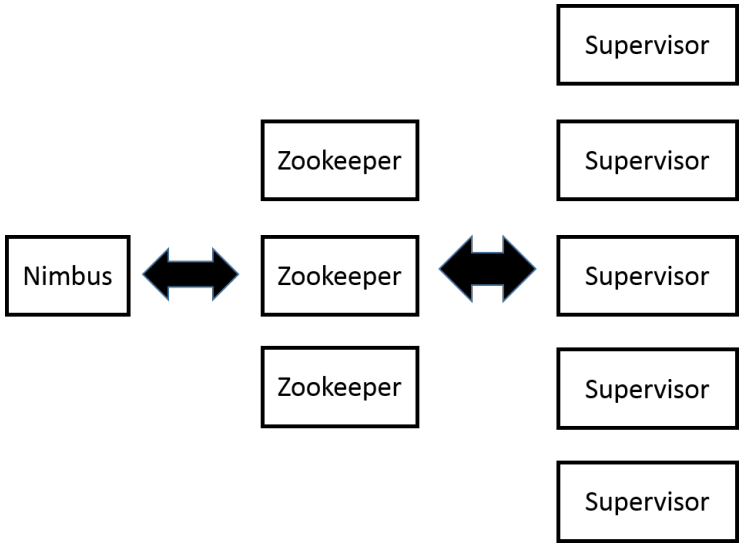


Figura 3.14: Cluster de Apache Storm

Lo que hace que esta herramienta sea interesante es su enfoque en la tolerancia a fallas y en la gestión. Implementa un procesamiento de mensajes garantizado de modo que cada tupla es completamente procesada mediante la topología, si se descubre que una tupla no ha sido procesada, se reproduce automáticamente desde el spout. También implementa la detección de fallas al nivel de la tarea. Cuando falla una tarea, los mensajes se reasignan automáticamente para reiniciar rápidamente el procesamiento. Incluye una gestión de procesos más inteligente que Hadoop, donde los procesos son gestionados por supervisores para garantizar que los recursos sean utilizados adecuadamente [64].

3.7. Apache S4

En esta sección hablaremos sobre Apache S4, otra herramienta para el procesamiento de datos en tiempo real. Veremos los conceptos y el funcionamiento de esta plataforma.

3.7.1. Introducción

S4 (las cuatro «s» de Simple Scalable Streaming System) es una plataforma de propósito general, distribuida, escalable, que permite el desarrollo de aplicaciones para el procesamiento de datos a gran escala en tiempo real [65]. Es de código abierto, está escrita en Java, fue creada por Yahoo S4 para tomar decisiones sobre la elección y el posicionamiento de anuncios pero posteriormente se ha visto su utilidad para el tratamiento arbitrario de corrientes de eventos.

3.7.2. Componentes

S4 ofrece una plataforma distribuida de tiempo de ejecución que se encarga de la comunicación, programación y distribución a través de contenedores, los cuales son llamados nodos S4. Los nodos son desplegados en el cluster S4 y el tamaño del cluster depende de las particiones lógicas siendo este fijo.

Las aplicaciones S4 son construidas en forma de grafo con elementos de proceso (PE) y canales de flujos que conectan esos PE. Los PE se comunican de manera asíncrona mediante el envío de eventos por el canal de flujo. Los eventos son enviados a los nodos de acuerdo a su clave. También es posible tener canales de flujo externos para recibir o enviar datos [66].

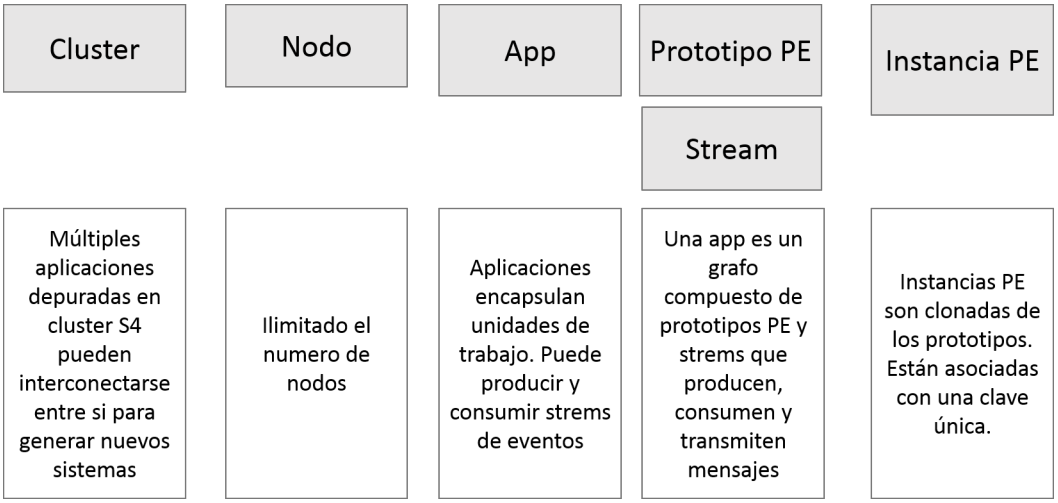


Figura 3.15: Arquitectura de Apache S4 [65]

3.8. S4 vs Storm

Se muestra la comparación [67] de estas dos herramientas de procesamiento de datos para tiempo real vistas en las secciones 3.6 y 3.7.

S4	**Storm**
Conceptualmente más potente	Garantiza procesamiento
Programación más sencilla	Mucho trabajo para el desarrollador
Recuperación del estado	Alto rendimiento
Balanceo de carga automático	No tiene balanceo de carga automático
Construido comunicación con cliente	Mayor control
Configuración compleja	Soporta programación con threads
Procesamiento Opaco	Funcionalidades avanzadas: Trident, topologías transaccionales
Potencialmente pueden perderse datos	Ecosistema
Complicado de depurar	Debugging sencillo (Local Topology)
En incubación en Apache	Distribución automática de tareas
Comunidad poco activa	Comunidad muy activa

3.9. Conclusiones

Vimos algunas de las herramientas existentes para cada pieza de la arquitectura Big Data, ya sea para almacenar, recolectar o procesar, las cuales pueden trabajar en conjunto o de manera independiente para resolver situaciones.

Debido al paradigma Map Reduce que tiene Big Data, Hadoop ha tomado gran importancia en la actualidad. Si bien la instalación y configuración de Hadoop es compleja, existen distribuciones gratuitas como son Hortonworks y Cloudera que facilitan el uso de Hadoop a todo público. Hadoop actualmente es una solución que tiene como principal función al procesamiento batch. Es una solución idónea para el procesamiento de grandes conjuntos de datos.

Nos enfocamos en investigar con mayor profundidad las herramientas Storm y Kafka ya que dirigimos la realización del caso de estudio en la recolección y procesamiento de datos en tiempo real y no en el almacenamiento, administración o visualización de datos.

Apache Kafka es una herramienta que consta de una arquitectura simple. Tiene varios mecanismos para realizar una recolección eficiente, por ejemplo, el uso del offset no se realiza del lado del broker sino del lado del consumidor, agilizando las tareas del broker.

Apache Storm conforma una herramienta de procesamiento en tiempo real que tiene varias ventajas en cuanto a funcionamiento (tolerancia a fallas, alto rendimiento, entre otras), una mayor estabilidad y popularidad. En cambio Apache S4 no está del todo aceptada en el desarrollo de estas aplicaciones y se encuentra en la incubadora de Apache. La obtención de información sobre la herramienta es escasa y no cuenta con ejemplos de implementación.

Capítulo 4

Familiarización con las herramientas

En este capítulo se presentarán las pruebas que hemos realizado de algunas de las herramientas mencionadas en el capítulo 3, tales como Apache Storm y Apache Kafka. En la Sección 4.2 el enfoque estará dado en el uso de Hortonworks, una de las distribuciones de Hadoop. Veremos cómo los conceptos presentados en el capítulo 3 son utilizados en la práctica, presentando también qué grado de dificultad tiene configurar y utilizar un ambiente de desarrollo para dichas herramientas.

4.1. Introducción

Dado que uno de los objetivos del proyecto es poder brindar una solución de análisis de Big Data para procesamiento de datos en tiempo real acerca de un problema que detallaremos en el capítulo 5, se analizaron diversas herramientas, las cuales brindan la posibilidad de obtener un resultado redituable a la hora de implementar un caso de uso en la práctica.

Trabajaremos con el ambiente Hortonworks para ver las prestaciones que presenta, y con el sistema operativo Linux para armar un ambiente de desarrollo. Entre las herramientas que abordaremos se encuentran Apache Storm y Apache Kafka. La primera constituye uno de los principales artefactos del ecosistema de Hadoop para el procesamiento de grandes volúmenes de datos en tiempo real, permitiendo desarrollar funciones a medida. La segunda es una herramienta de recopilación de datos implementando un sistema de mensajería. Nos basaremos en estas herramientas ya que aplican para el procesamiento en tiempo real de datos a diferencia de las demás.

Hemos realizamos varios casos de prueba de ensayo propio para probar el funcionamiento y familiarizarnos con las herramientas y el ambiente de desarrollo.

4.2. Hortonworks

Hortonworks ha anunciado la disponibilidad de Hortonworks Sandbox, una máquina virtual flexible y sencilla de usar para aprender y explorar Apache Hadoop.

La solución se puede descargar de manera gratuita, es un sistema operativo Linux que contiene Hadoop pre-configurado, es portable y contiene un conjunto de tutoriales paso a paso que facilitan el aprendizaje de Hadoop por cuenta propia. Hortonworks Sandbox está diseñado para ayudar a estrechar el espacio entre la gente que desea aprender y evaluar Hadoop, y las complejidades de echar a andar una evaluación de cluster de Hadoop.

Se descargó la imagen de la máquina virtual Hortonworks Sandbox desde la página de Hortonworks [68] para poder utilizar algunas herramientas de Hadoop. Se configuró la máquina virtual con VMWare Player y VMWare Workstation en Windows 8.1 y en Windows 7, ambos programas para el uso de máquinas virtuales (también se puede configurar en VirtualBox).

Se puede descargar VMWare desde la página [69], e instalar en Windows o Linux.

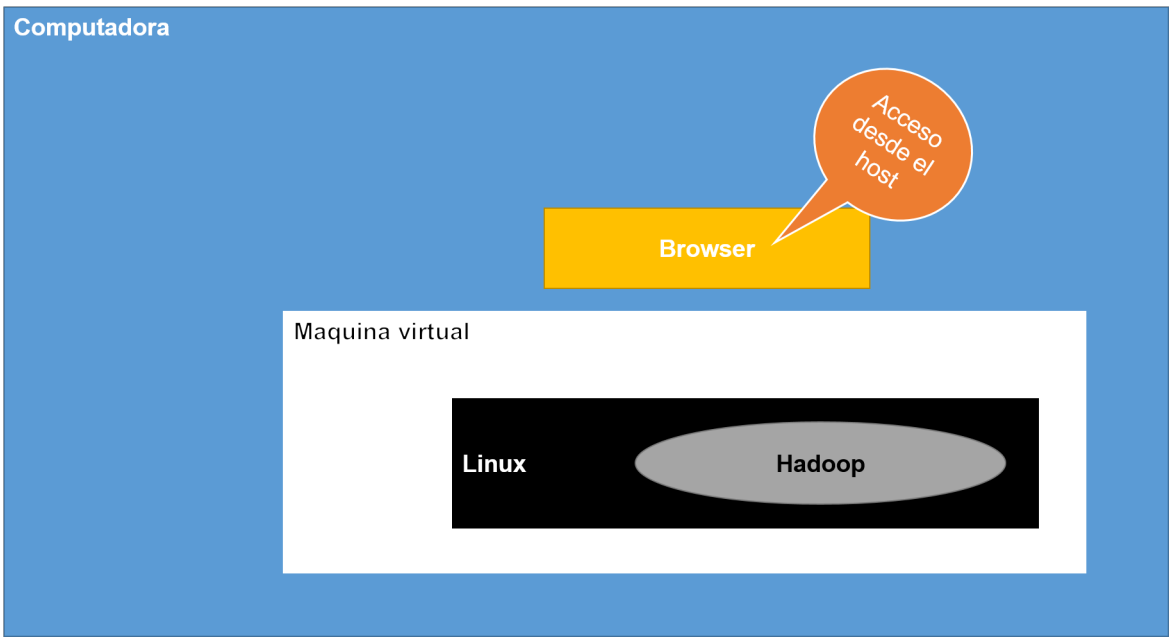


Figura 4.1: Esquema de Hortonworks Sandbox

La principal ventaja de este tipo de servicios es que ofrecen un entorno Hadoop de una manera sencilla y sin instalación. En unos pocos minutos se puede tener Hadoop operativo y todas las herramientas adheridas.

Una desventaja es que necesitan de un computador medianamente potente (memoria RAM mayor a 8gb y un CPU i5 o i7 aproximadamente). Esto lo podemos

apreciar al utilizar la herramienta Ambari para manipular los servicios Hadoop en un computador de menores recursos donde se nota que el sistema se enlentece y no responde.

Hortonworks contiene la consola Ambari como habíamos señalamos en la sección 3.3.1.1, la cual es una interfaz web amigable con la que podemos dar de alta, iniciar, finalizar o modificar servicios tales como Zookeeper, Storm, Kafka, HBase, Hive, entre otros. Se puede acceder a la consola desde un navegador web accediendo a la dirección «<dir_ip_maquinavirtual>:8080».

4.3. Familiarización con Apache Kafka

Como hemos visto, es un sistema de mensajería productor-consumidor, con lo que pudimos comprobar su funcionamiento descrito en el marco teórico, pudiendo identificar los conceptos aplicados en la práctica. Esta herramienta al tener un productor-consumidor en tiempo real, se puede utilizar para soluciones en las que se generen datos en tiempo real y poder realizar su procesamiento en todo momento.

4.3.1. Caso de prueba en Linux

Se ha instalado y configurado Kafka para probar la herramienta en modo local en un ambiente con sistema operativo Linux (Ubuntu 14.04).

Hemos ejecutado en una consola (línea de comandos) el inicio del servidor Zookeeper, luego el broker, un productor y en otra línea de comandos un consumidor, comprobando las funcionalidades de publicar y leer los mensajes de forma exitosa.

Posteriormente se ha realizado una aplicación que contiene un productor el cual envía una cantidad n de mensajes al broker, para luego ser obtenidos por el consumidor. Como vimos la metodología de trabajo de Kafka se basa en consumidor-productor, para generar un sistema aplicando esta herramienta basta con implementar un productor y un consumidor, y crear un topic en donde irán los mensajes que se envían al broker.

Finalmente modificamos el productor para realizar una simulación de eventos en base a los datos brindados por la empresa UnoWiFi que serán analizados en el capítulo 5, el cual lee datos y los envía al broker en diferentes intervalos de tiempo para luego obtenerlos desde un consumidor.

4.3.2. Caso de prueba con Hortonworks

Mediante el ambiente de desarrollo Hadoop llamado «Hortonworks» visto en la sección 4.2 hemos probado el segundo caso de prueba. Este caso de prueba consta de un productor que recopila información sobre el identificador y nombre de un conductor de un camión y la geolocalización del camión combinándolos con eventos en tiempo real que reportan los excesos de velocidad, desvíos, tramos inseguros de las rutas de New York, Estados Unidos, y un consumidor que consume del broker para mostrar esos datos en tiempo real.

4.3.3. Apache Kafka vs otros servicios de mensajería

4.3.3.1. Introducción

En esta sección presentaremos un estudio realizado por LinkedIn en el cual evalúan la performance de la herramienta Apache Kafka contra dos herramientas de mensajería tradicionales [70].

4.3.3.2. Estudio

El experimento conducido por LinkedIn consistió en comparar Apache Kafka con Apache ActiveMQ versión 5.4 (un broker de mensajes de código abierto, actúa como mediador de mensajes entre aplicaciones emisoras y receptoras, proporciona comunicación asincrónica entre aplicaciones) [71] con almacenamiento en KahaDB y RabbitMQ versión 2.4 (también es un broker de mensajes de código abierto, se encuentra dentro de la categoría de middleware de mensajería)[72].

El experimento fue ejecutado en dos computadoras con sistema operativo Linux, con 8 cores de 2Ghz, 16Gb de memoria RAM, 6 discos RAID10 y ambas conectadas en red por un vínculo de 1GB de velocidad. Una de estas computadoras cumplía el rol de broker y la otra de productor o consumidor. Se configuró el broker en todos los sistemas de mensajes de forma asíncrona con el sistema de almacenamiento.

Se ejecuta un simple productor que publica un total de diez millones de mensajes, cada uno de 200bytes. Los productores de Kafka envían los mensajes en lotes de 1 a 50 mensajes. Las otras dos herramientas envían en lotes de 1 mensaje.

Algunas de las razones por las que Kafka tiene mejor performance que las otras herramientas analizadas es que:

- Kafka no espera respuesta de reconocimiento desde el broker, por lo que envía mensajes lo más rápido posible que el broker maneja.

- Tiene un formato de almacenamiento más eficiente. Cada mensaje en Kafka tiene una cabecera de 9 bytes mientras que ActiveMQ tiene de 144bytes, es decir que ActiveMQ utiliza 70 % más de espacio.

En cuanto al análisis de los consumidores tenemos que se utiliza un consumidor solo para recuperar el total de los 10 millones de mensajes. Se configuraron todos los sistemas para que la solicitud de extracción precapture aproximadamente la misma cantidad de datos, hasta mil mensajes o alrededor de 200kb. Para ActiveMQ y RabbitMQ se establece de modo automático.

Algunos de los aspectos que hacen que se tenga una mejor performance son:

- Se tiene un mejor sistema de almacenamiento. Menos bytes son transferidos del broker al consumidor en Kafka.
- El broker de ActiveMQ y RabbitMQ tienen que mantener el estado de entrega de cada mensaje, y algunos hilos de ejecución se ocupan de escribir páginas KahabDB a discos.
- Kafka reduce la sobrecarga de transmisión mediante la API sendfile.

4.4. Familiarización con Apache Storm

4.4.1. Caso de prueba con Hortonworks

Este caso de prueba se llevó a cabo sobre el ambiente de desarrollo Hortonworks, el cual ejecuta sobre un sistema operativo Linux llamado «CentOS» y tiene integrado varias de las herramientas Hadoop donde es posible instalar Storm, entre otras. Hemos instalado, configurado y ejecutado un ejemplo brindado por el mismo creador, Nathan Marz, que muestra el funcionamiento del procesamiento de datos en tiempo real, el cual es un proyecto Maven de código abierto en lenguaje de programación Java y Python llamado «storm-starter» [73].

Este proyecto permite ver cómo son implementados los conceptos de esta herramienta y su comunicación entre ellos para formar una topología. Existen varias funcionalidades en este trabajo. Nosotros nos enfocaremos en comentar la función llamada «WordCountTopology», la cual se basa en contar la cantidad de veces que se repite una palabra en un flujo de frases. Con este estudio bastará para observar cómo se combinan los bolts y spouts para implementar una topología. Storm nos brinda todas las primitivas necesarias para poder construirla de manera eficiente.

En este caso todo el problema de resolver la cantidad de palabras se resuelve en una topología.

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("spout", new RandomSentenceSpout(), 5);

builder.setBolt("split", new SplitSentence(), 8).shuffleGrouping("spout");
builder.setBolt("count", new WordCount(), 12).fieldsGrouping("split", new Fields("word"));
```

Figura 4.2: Fragmento de código de una topología en Storm

La idea de desarrollar aplicaciones de Storm es definirse una topología que contiene bolts y spouts trabajando en conjunto para poder resolver un problema planteado.

En el fragmento de código de la Figura 4.2 tenemos que se desea crear una nueva topología llamada «builder». A esta se le configura un spout con un identificador llamado «spout» (el cual sirve de autoreferencia) que genera aleatoriamente un conjunto de frases cada 100ms. Posteriormente tenemos la configuración de dos bolts.

Una vez ejecutada la aplicación empezará a consumir datos y procesarlos. La ejecución no terminará hasta que el programador lo indique.

En el ejemplo de la figura 4.2 vemos que se define un spout que genera un conjunto de frases cada 100ms tiempo para luego ser consumidos por dos bolts seguidos. Uno realiza la función split para obtener cada palabra de la frase y otro para contar la cantidad de palabras repetidas.

Para ver el resultado bastará con ver el log de los workers, como se puede apreciar en la figura 4.3. Aquí se observa claramente que se están contando las palabras de las frases. En primera instancia se realiza un «split» y luego se incrementa la cantidad de veces de esa palabra. En este caso se generaron dos workers, que como hemos visto en capítulos anteriores, los workers son quienes se encargan de ejecutar el procesamiento, siendo controlados por un supervisor qué interactúa entre ellos.

```
10:14:03 b.s.d.task [INFO] Emitting: split default ["doctor"]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:22, stream: default, id: {}, [away]
10:14:03 b.s.d.task [INFO] Emitting: count default [away, 4084]
10:14:03 b.s.d.task [INFO] Emitting: split default ["away"]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:17, stream: default, id: {}, ["away"]
10:14:03 b.s.d.task [INFO] Emitting: count default [away, 4085]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:22, stream: default, id: {}, [am]
10:14:03 b.s.d.task [INFO] Emitting: count default [am, 4226]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:20, stream: default, id: {}, [am]
10:14:03 b.s.d.task [INFO] Emitting: count default [am, 4227]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:22, stream: default, id: {}, [two]
10:14:03 b.s.d.task [INFO] Emitting: count default [two, 4226]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:22, stream: default, id: {}, [with]
10:14:03 b.s.d.task [INFO] Emitting: count default [with, 4226]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:22, stream: default, id: {}, [apple]
10:14:03 b.s.d.task [INFO] Emitting: count default [apple, 4086]
10:14:03 b.s.d.executor [INFO] Processing received message source: split:20, stream: default, id: {}, [two]
10:14:03 b.s.d.task [INFO] Emitting: count default [two, 4227]
```

Figura 4.3: Fragmento del log de salida de Storm

Se puede acceder a una interfaz de usuario llamada Storm UI desde un navegador web donde se muestra la configuración de Storm, que topologías se tiene cargadas y se detalla el estado de los supervisores y de los workers como se puede observar en la Figura 4.4.

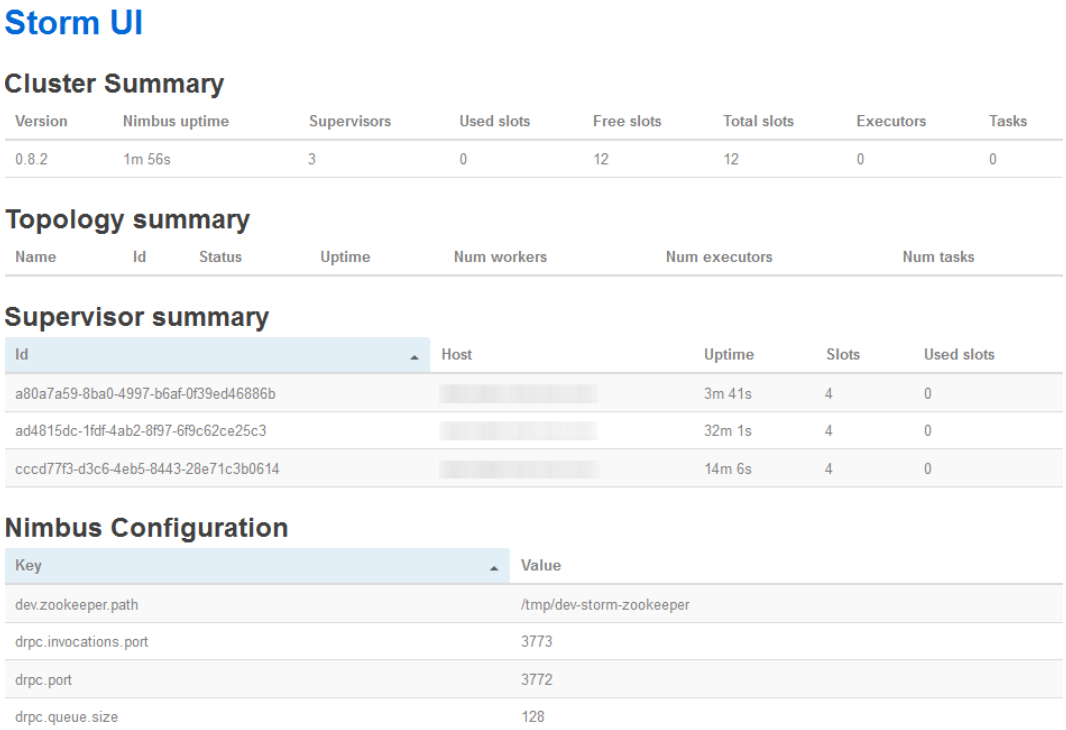


Figura 4.4: Storm UI

4.4.2. Caso de prueba en Linux

Otra alternativa llevada a cabo fue instalar Storm en un sistema operativo Linux desde cero, en el cual se necesitaron hacer varias pre-instalaciones para poder utilizarlo correctamente. Alguna de esas pre-instalaciones fueron: Zookeeper y ZeroMQ como vimos en la sección 3.6 ya que la herramienta requería de las mismas para funcionar y Maven para poder compilar y ejecutar aplicaciones Storm. Para la configuración se deben configurar los puertos de Zookeeper para ser conectados con Storm, el puerto de acceso a Storm UI.

Se realizó la ejecución del ejemplo del caso de prueba anterior 4.4.1, siendo necesario dar de alta el Supervisor, Nimbus, Storm UI y el servidor Zookeeper.

4.5. Apache Kafka - Apache Storm

Kafka y Storm pueden trabajar en conjunto para realizar una solución. Como sabemos Storm construye sus topologías en base a bolts y spout, y Kafka basa su solución en productor/consumidor, por lo que existen los llamados Kafka Spout, que son el vínculo entre ambos para generar una solución híbrida. Este nuevo elemento funciona como consumidor y como spout a la vez, es decir es un spout que consume los mensajes de broker para luego enviarlos a algún bolt en donde se procesan.

4.5.1. Caso de prueba con Hortonworks

Utilizando nuevamente el ambiente de desarrollo Hadoop «Hortonworks» se realizó un caso de prueba para ver cómo funcionan en conjunto ambas herramientas.

Basándonos en el caso de prueba visto en 4.3.2 que genera información acerca de los camiones a partir de los eventos y rutas es que se probó la conjunción de ambas herramientas.

Se utiliza el productor del caso de prueba mencionado que envía información de los eventos, rutas y camiones al broker. El consumidor kafka-spout consume esos mensajes del broker con el fin de procesarla aplicando Storm. Básicamente se implementó una topología que retorna la información ordenada que llego al spout.

Un vez ejecutando la topología y con el productor Kafka activo es que se empieza a procesar los datos en tiempo real. Mediante la UI de Storm se pueden observar el spout y los bolts que están ejecutándose. Se accede a toda la información en cuanto a gestión y estado de la topología del ambiente Storm refiere. En la figura 4.5 podemos apreciar lo señalado anteriormente.

Por otro lado existen «logs» que son archivos que contienen información de los workers e imprimen lo que se desea. En este caso es el resultado ordenado que retornan los bolts.

4.5.2. Caso de prueba en Linux

El siguiente caso de prueba se realizó en un sistema operativo Ubuntu configurando algunas herramientas Hadoop manualmente para armar un ambiente de desarrollo para Kafka y Storm.

En este caso hemos instalado Zookeeper, ZeroMQ, Maven, Java, entre otros, tal como comentamos en el caso de prueba 4.4.2 y Apache Kafka. Una de las configuraciones a tener en cuenta es la concordancia en cuanto a las direcciones IP y puertos que se utilizan para la conexión entre ambas herramientas.

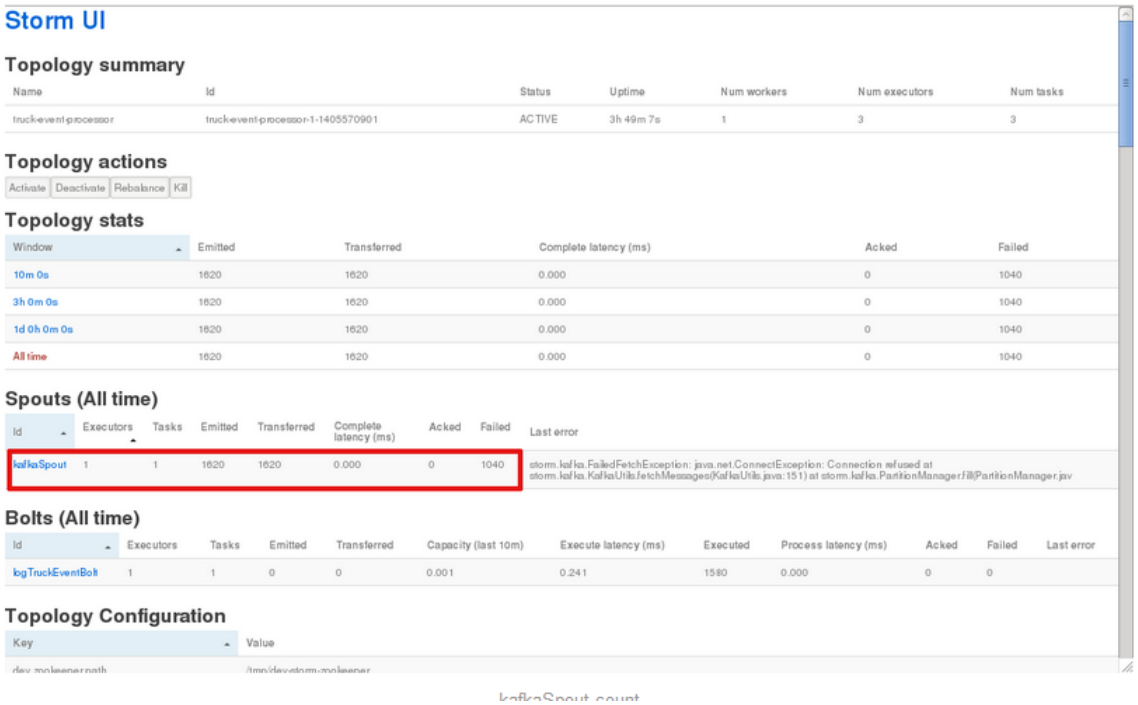


Figura 4.5: Imagen de Storm UI en la topología

En principio se probó un simple caso enviando mensajes desde el productor, el consumidor/spout obtiene esos mensajes y los traslada a un bolt que simplemente muestra el mensaje. Con este estudio comprobamos el funcionamiento de ambas herramientas. Se amplió el caso enviando miles de mensajes (cada mensaje corresponde a una tupla) numerados desde Kafka para probar si efectivamente no hay pérdida de mensajes y se apreció que esto es correcto. La tolerancia a fallas es garantizada ya que desde Storm se obtuvo la cantidad de mensajes enviadas observando el archivo de logs de los workers.

4.6. Conclusiones

A la hora de implementar los casos de prueba se apreciaron los conceptos y arquitecturas de las herramientas presentadas en el capítulo 3. Además se pudo observar su aplicación en un caso práctico como es el monitoreo de vehículos en tiempo real.

En cuanto a la distribución Hortonworks, es un ambiente Hadoop suficientemente completo para el desarrollo de soluciones de Big Data al contener la gran mayoría de las herramientas del ecosistema instaladas y configuradas. Una de las desventajas es que requiere un computador con buenos recursos en cuanto a memoria RAM y CPU, y no cuenta con una interfaz gráfica para poder implementar soluciones de manera amigable.

Sobre Kafka observamos una fácil instalación, configuración e implementación de soluciones en modo local. Se prueba la eficacia de la mensajería, enviando mensajes y llegando a destino sin pérdidas y en orden.

En el caso de Storm su instalación y configuración resulta un tanto compleja ya que depende de varias herramientas. La implementación no es tan trivial como en el caso de Kafka. Requiere de más dedicación al armar una topología ya que es donde se realiza el procesamiento de los datos. Se ha probado que la tolerancia a fallas es una cualidad correcta por lo menos en lo que respeta a modo local.

Apreciamos que las herramientas pueden ser utilizadas independientemente o sino formar una solución híbrida entre ambas para tener una arquitectura distribuida. Esta conjunción tiene sus dificultades al configurar el vínculo entre ellas. Una vez generado el vínculo entre ambas herramientas, la implementación se puede descomponer en dos partes (implementar el productor Kafka e implementar el procesamiento en Storm).

A lo largo de este capítulo mostramos como nos familiarizamos con las herramientas y como generamos un ambiente para el desarrollo y ejecución de una solución del caso de estudio que se presenta en el 5.

Capítulo 5

Caso de estudio

En este capítulo se presenta la especificación y el análisis del caso de estudio enfocado en un problema real. En la Sección 5.1 se introduce brevemente la relación entre los objetivos del proyecto con el caso de estudio. En la Sección 5.2 se presenta el problema planteado. Finalmente en la Sección 5.3 se plantea como se analizó y la metodología de resolución del problema de estudio.

5.1. Introducción

Uno de los objetivos del proyecto es poder emplear los conceptos y herramientas aprendidas sobre el procesamiento de datos masivos en tiempo real en un caso concreto. Para ello nos contactamos con Gustavo Azambuja, usuario responsable y director de la empresa UnoWiFi quien nos propuso un problema real que desea resolver. El objetivo es, empleando los conceptos y herramientas de Big Data de procesamiento en tiempo real, poder brindarle un prototipo que genere resultados que le sirvan de ayuda en su emprendimiento.

UnoWiFi es una empresa que distribuye puntos de acceso inalámbricos en locales adheridos. Su misión es brindar a los comercios que contratan el servicio la posibilidad de que los clientes que se conecten a la red puedan acceder a Internet y a diversa información del local, como pueden ser ofertas, promociones, publicidad, etc.

El caso de estudio es un problema real que requiere contar la cantidad de personas que se encuentran dentro de establecimientos comerciales que son cliente de UnoWiFi.

El saber la cantidad de personas en todo momento resulta ser información útil para los negocios ya que permite identificar cuáles son los horarios pico en que hay mayor o menor cantidad de clientes, poder identificarlos al acercarse al local,

obtener estadísticas del perfil de los mismos, básicamente, mejorar el funcionamiento de los negocios a partir del monitoreo y ofrecer más herramientas para la toma de decisiones en tiempo real.

5.2. Caso de Estudio

5.2.1. Presentación del caso de estudio

Trabajamos con Gustavo Azambuja como usuario responsable, quien colaboró generosamente con nosotros en todo momento. Nos explicó la situación actual de UnoWiFi y nos planteó posibles problemas a resolver en tiempo real, también nos brindó los datos para entender y resolver el caso de estudio.

La realidad propuesta se basa en routers inalámbricos instalados en diversos locales comerciales de distintos rubros, como por ejemplo: restaurantes, supermercados, shoppings, etc., los cuales brindan acceso a Internet a los visitantes que disponen de dispositivos móviles y computadoras portátiles. En base a las conexiones a Internet de los dispositivos se recopilan datos de los usuarios y de los dispositivos que luego pueden ser analizados. El sistema le da la opción al usuario de registrarse e ingresar sus datos activamente, por lo que de esa forma se obtienen más datos del usuario de los que se obtienen de forma pasiva a través de la conexión del dispositivo.

Es importante señalar que un principio el problema planteado no representaba un problema de Big Data en tiempo real ya que, por un lado consideramos que los datos brindados no eran de un volumen lo suficientemente grande como para enmarcarlo dentro del paradigma de Big Data y por otro lado no se contaba con un flujo de datos en tiempo real sino que los datos son estáticos. A raíz de esto lo que se hizo fue una simulación de recopilación de datos en tiempo real, realizándose una implementación de una solución escalable.

Para extender el caso de estudio y transformarlo en un problema de Big Data se consideró una red más extensa de la que hay en la actualidad de routers WiFi distribuidos en diversos puntos recopilando datos constantemente y almacenándolos en un servidor.

5.2.2. Especificación del caso de estudio

Un local cuenta con uno o varios routers WiFi que brindan acceso a Internet a los clientes, a los que llamaremos sensores. Cada sensor genera una conexión con un dispositivo que brinde conexión Wireless. Cuando un dispositivo (celular, tablet, computadora) tiene la opción de conexión WiFi activada cada cierto tiempo obtiene

las redes que se encuentran disponibles para conectarse. Por otro lado los routers reciben datos de los dispositivos que ven la red. Recopilan información tal como: MAC del dispositivo, fecha y hora de conexión, fecha y hora de desconexión, coordenadas del sensor, potencia de señal, entre otras, lo que genera un evento. Todo basado en datos anónimos pero brindando la posibilidad de que el cliente que quiera pueda registrarse y por lo tanto pueda ser identificado. Para identificar al cliente, el mismo podrá registrarse para así asociarse con su dispositivo a la red y obtener acceso a Internet. Cada un minuto el router envía toda la información recolectada en ese tiempo (dispositivos que ven la red) a un servidor web donde:

- En caso que no exista ese dispositivo y sensor se agrega a la base de datos guardando la información anteriormente señalada.

- En caso que exista ese dispositivo se actualiza la fecha de desconexión en la base de datos para saber cuándo se conectó y se desconectó el dispositivo. Por ejemplo, si un dispositivo ya guardado en la base es detectado, en la siguiente intervención del router (cada minuto) no se agregará un nuevo registro a la base, sino que se actualizará la hora y fecha de desconexión para el registro que ya existía. De esta manera siempre se está actualizando el mismo registro. Se agregará un nuevo registro a la base cuando el dispositivo sea detectado en una lectura del router que difiera del minuto con la fecha y hora de desconexión, en relación al último registro para ese dispositivo.

La base de datos es MySQL y contiene: el dispositivo, sensor, señal, coordenadas, fecha de conexión y desconexión de ese dispositivo en ese sensor y tiempo de conexión. El conjunto de datos entregados pertenecen a un rango de fechas (15 de diciembre 2014 hasta el 15 de febrero 2015).

Actualmente para obtener la cantidad de personas en determinadas fechas UnoWiFi realiza una simple consulta en MySQL. En la base de datos se tiene el tiempo de permanencia (ya que cada minuto se van actualizando los valores de los dispositivos que mantienen la conexión). La consulta consiste en realizar una selección de registros según la fecha interesada, con el condicional que esa fecha sea mayor a la fecha de conexión y menor a la fecha de desconexión. La dificultad que surge con esta forma de resolver el problema es que la idea es obtener el resultado en tiempo real, por lo que teniendo en cuenta que se piensa escalar de gran forma en cuanto a la cantidad de sensores y dispositivos conectados, lo que generarían grandes cantidades de datos, la solución con herramientas tradicionales se torna inviable.

5.3. Análisis del Caso de Estudio

El análisis del caso de estudio se realizó en las siguientes etapas: primero la interpretación de los datos recibidos, de modo de conocer que información tenemos en esos datos para luego determinar de qué manera podemos procesarlos de forma de obtener el resultado que se está buscando.

5.3.1. Interpretación de los datos

Nuestro primer paso fue analizar los datos entregados, los cuales estaban contenidos en un archivo de tipo JSON, CSV y un script SQL. Observamos los distintos campos y cuáles podrían ser sus respectivos valores, combinándolos para derivar otros datos de importancia. Concluimos que los datos que se obtienen y que son relevantes son:

- Clientes: la información proveniente de los clientes (identificador del dispositivo, nombre, teléfono, sexo, edad, entre otros), en el caso de que estos deseen ser identificados.

- Dispositivos: son los dispositivos tanto smartphone, tablets u computadoras, donde cada uno de estos cuenta con una dirección MAC que es única para cada dispositivo, identificador del primer sensor que se conectó, del último sensor que se conectó, entre otros.

- Sensores: estos representan a los routers conectados a la red, donde se encuentran ubicados, sus coordenadas y direcciones MAC que los identifican.

- Eventos: representan las conexiones y desconexiones que existen entre los dispositivos y los sensores, brindan la fecha y hora que se conectó, tiempo de permanencia, etc.

Para nuestro problema los datos que nos parecieron interesantes eran únicamente los de eventos, ya que contenían toda la información (identificador del dispositivo, identificador del sensor, potencia de señal, fechas de conexión y desconexión) que se necesita para resolver el problema.

En la etapa de análisis nos dimos cuenta que estos datos no podían ser tratados como datos en tiempo real al ver que ya contenían el tiempo de permanencia de cada dispositivo en cada conexión por lo que se pensó en una simulación que explicaremos en la sección 5.4. La idea básicamente es analizar los datos en tiempo real, es decir obtener los dispositivos que reconoce el sensor cada minuto.

5.3.2. Procesamiento de datos

En esta etapa es donde se resuelve la lógica para procesar los datos que se tengan recolectados, de manera de obtener el resultado. Si analizamos el problema de obtener qué cantidad de personas hay en un local y lo intentamos resolver analíticamente, consta de obtener los datos, analizarlos y filtrarlos según comparaciones de fecha de conexión y desconexión con la fecha actual hasta llegar al resultado. Para esto hay que tener en cuenta que la red de WiFi es privada pero cualquier dispositivo puede ser capaz de identificarla aunque no esté conectado. Un dispositivo que tenga conexión Wireless puede reconocer todas las redes WiFi que existan, y a su vez esta red WiFi obtiene información del dispositivo por lo que existen dificultades para obtener una solución exacta.

Debido a que los actuales clientes de UnoWiFi son establecimientos que están en la ciudad, es coherente pensar que no todas las personas que son detectadas por los sensores que se tienen instalados en el lugar sean clientes del mismo, en este sentido surgen las siguientes dificultades a tener en cuenta:

- Puede ser que hayan personas fuera del local con un dispositivo que reconozca la red. En este caso están:
  - Las personas que pasan por el local, es decir que la red descubre el dispositivo y al poco tiempo desaparece (peatones, conductores, pasajeros, etc.).
  - Personas vecinas al local.
  - Personas que se encuentran detenidas en el exterior del local, por ejemplo esperando un ómnibus.
  - Empleados del local.
- Personas que no cuenten con dispositivos inteligentes o no tengan activado las conexiones WiFi de su dispositivo.

Luego de entrevistarnos con el usuario responsable para intercambiar ideas, conocer sus dificultades y el resultados que deseaba obtener sabiendo que no se puede llegar a un valor exacto debido a las dificultades planteadas anteriormente, se establecieron tres puntos:

- Contar un dispositivo como activo si paso más de 5 minutos conectado, para el caso de dispositivos que solo pasan por el local.

- Contar un dispositivo como activo si su última conexión está en un rango de 25 minutos. Este punto sirve para contemplar el hecho de que hay dispositivos que permanecen en el local y perdieron la señal por un tiempo.
- Tener en cuenta la potencia de la señal de la conexión, puede ser un indicador de distancia. Se puede utilizar para saber si se encuentra en el establecimiento o no.

Estos puntos fueron tenidos en cuenta en el momento de pensar una solución al problema, ya que forman parte del procesamiento de la solución, el decidir si un dato cumple las reglas para considerarse que se encuentra dentro del establecimiento o no.

Flujo a seguir si lo resolviéramos analíticamente sin importar la herramienta que usemos, ya sea de Big Data o no, sería:

1. Obtener los datos. Estos llegarían a los módulos encargados del procesamiento desde el router, mediante un servicio por el cual el router y el módulo de procesamiento se puedan conectar. El envío de esta información por parte del router se realizaría inmediatamente después de que envíe la misma a ser guardada en la base de datos. En nuestro caso, al tener que simular el flujo de los datos en tiempo real, los datos serán leídos desde una fuente estática.
2. Filtrar los datos según sensor, para tener los datos diferenciados por sensor. Este filtrado es posible ya que la información que es recibida contiene el identificador del sensor.
3. Filtrar los que no han desaparecido aún, serían las personas que siguen conectadas. Este filtrado se realiza comparando el registro recibido con el de nuestra base de datos; en caso de que ya no esté se descarta.
4. Filtrar todos aquellos dispositivos que se hayan conectado antes del horario actual y permanezcan conectadas por más de 5 minutos para evitar conexiones insignificantes.
5. Filtrar cuáles son posibles clientes del local y cuáles no. Esto se determina considerando dos aspectos: su tiempo de permanencia en la red y verificando que el individuo no figure en una base de datos auxiliar que contiene a las personas que no son clientes del establecimiento.
6. Obtener el conteo de la cantidad de personas aproximadamente (no se puede obtener un resultado exacto debido a las dificultades planteadas). Se guardará

en estructuras de fácil acceso a la cantidad de personas que se consideren clientes del local, de manera de acceder de forma rápida a ese valor para su actualización.

El flujo establecido se determinó de acuerdo a las dificultades identificadas en un principio del análisis. La dificultad planteada es traducir ese flujo al paradigma de las herramientas de Big Data que utilizamos para resolver el problema.

5.4. Solución

La resolución del problema está implementada en base a una fusión de Kafka-Storm para representar la obtención y procesamiento de datos en tiempo real. Se eligieron estas herramientas para tener una arquitectura compleja y mostrar como Kafka y Storm (dos plataformas de propósito diferente) pueden trabajar en conjunto.

En primera instancia se empezó a desarrollar en un ambiente Hortonworks ya que provee muchas herramientas de Hadoop para simplificar instalaciones y configuraciones complejas, pero se desistió de utilizarlo debido a que nuestra infraestructura escasea de los requerimientos de hardware, tornándose lenta la ejecución de las herramientas. Decidimos entonces instalar un ambiente de desarrollo desde cero utilizando sistema operativo Ubuntu, instalando y configurando manualmente las herramientas que eran necesarias (vistas en el capítulo 4).

Al analizar el problema y los datos con los que contábamos nos encontramos con que el problema no era completamente de Big Data en tiempo real por lo que se realizaron pasos previos para obtener un ambiente ideal. Por un lado el volumen de los datos resultó no ser suficiente y por otro lado el contenido de ellos no representaba una recolección en tiempo real. Eran datos estáticos, debido a que contenían fecha de conexión y fecha de desconexión de cada dispositivo, dado que cada un minuto los sensores envían los datos obtenidos y se actualizaba la base de datos, por lo que se descompusieron esas fechas mediante procedimientos MySQL para generar una simulación de datos en tiempo real. Por ejemplo, si tenemos un registro que para un dispositivo tiene fecha de conexión 0 y fecha de desconexión 10, se generan diez registros con diferencia de 1 minuto entre ambas fechas. Otra consideración fue trasladarse a la fecha del primer evento y simular un reloj propio.

Luego de pulir esos datos se realizó una simulación para resolver la situación de recopilar eventos en tiempo real. Si bien es una simulación, fácilmente se puede modificar para un futuro caso en que los datos que se obtengan sean en tiempo real, el cual abordaremos en la sección 5.5.

La simulación consta de leer línea por línea un archivo CSV, el cual contiene los registros previamente regenerados en base a los datos reales donde cada registro contiene: identificador del dispositivo, identificador del sensor, fecha de conexión, fecha de desconexión, potencia de señal, entre otros. Apache Kafka (vista en el capítulo 3.5) es una herramienta de recolección de datos que también nos permite realizar esta simulación. Kafka utiliza una arquitectura de productor-consumidor, en este caso nuestro productor es quien hace la lectura del archivo CSV para enviar al broker, y el consumidor es quien consume los registros que llegan al broker para luego procesarlos. El consumidor es el vínculo entre Kafka y Storm, ya que puede trabajar como consumidor y como spout (componente de Storm) a la vez, se denomina como «kafka-spout». Entonces nuestro consumidor será un kafka-spout que consuma esos datos para enviarlos a un bolt. El bolt (componente de Storm) es quien hace todo el procesamiento de esos registros (más conocidos como tuplas en Storm) como podemos apreciar en la Figura 5.1.

Por ejemplo, el productor lee el evento y lo envía al broker (auspicia de buzón), una vez en el broker el consumidor consume ese evento y se lo envía al bolt quien es el motor del procesamiento, es decir se verifica si cumple con lo establecido para considerarse que se encuentra adentro del establecimiento (comparaciones de fechas, potencia de señal).

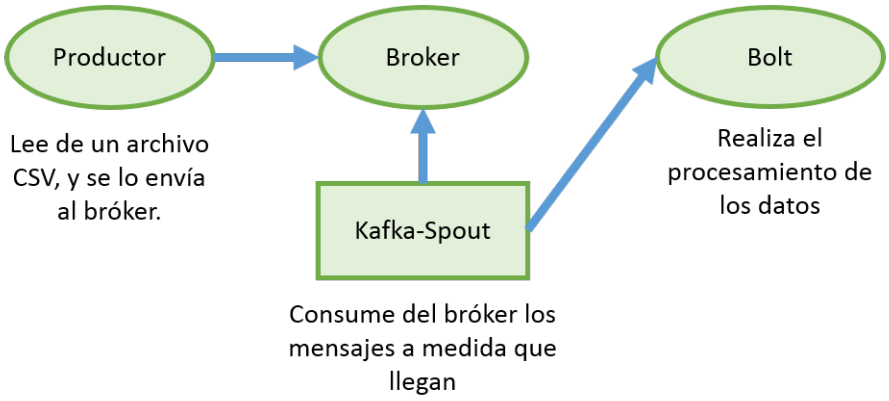


Figura 5.1: Diagrama de solución.

El procesamiento consiste en filtrar los datos para obtener el resultado esperado. Es decir, dado un evento determinar si cumple que la fecha de conexión es mayor a 5 minutos, y si la fecha de desconexión es menor a 25 minutos, también si la potencia de señal es mayor al promedio de las potencias de señal (este valor puede ser definido por el sensor).

Al utilizar Storm, éste basa las funcionalidades en la construcción de una topología que represente el tratamiento de los datos para llegar a un resultado. En nuestro

caso con diseñar una topología que resuelvan los filtros que hay que aplicar bastará. Veremos al bolt como las funciones de filtrado y el spout integrado a kafka como el recolector de datos.

Una vez culminada la etapa de implementación se procedió a ejecutar la solución y ver que los resultados sean coherentes.

5.4.1. Pruebas

En esta sección se muestran algunas pruebas que se realizaron para verificar la implementación. Las mismas tienen el objetivo de mostrar al lector que se obtienen resultados correctos, no existen pérdidas de datos y se tienen tiempos de latencia bajos.

Una prueba fue comparar los resultados de nuestra solución con los de una consulta en MySQL. Los datos que nos enviaron ya contenían el tiempo de conexión de cada dispositivo en cada sensor, por lo que se cargaron en una base de datos MySQL y se realizó una consulta para obtener cuántos dispositivos existían conectados para un determinado sensor, y observamos en la Figura 5.2 que ambos resultados coincidían.

```
q.w.a.DeviceCountBolt [INFO] Hoy:2014-12-12 23:50:28 -Inicio: 2014-12-12 18:51:41diff: 17927
q.w.a.DeviceCountBolt [INFO] Hoy:2014-12-12 23:50:28 -Inicio: 2014-12-12 18:39:58diff: 18630
q.w.a.DeviceCountBolt [INFO] Hoy:2014-12-12 23:50:28 -Inicio: 2014-12-12 17:02:13diff: 24495
q.w.a.DeviceCountBolt [INFO] Hoy:2014-12-12 23:50:28 -Inicio: 2014-12-12 18:38:04diff: 18744
q.w.a.DeviceCountBolt [INFO] idSensor:72 - Cant:4
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	idDevice	pwd	date	connected	traffic	disappear	diffTime	idSensor	coordinates
174...	633386	77	2014-12-12 17:02:13	NULL	0	2014-12-15 ...	239522	72	-34.9015745,...
174...	1420516	77	2014-12-12 18:38:04	NULL	0	2014-12-15 ...	233781	72	-34.9015745,...
174...	686923	76	2014-12-12 18:39:58	NULL	0	2014-12-15 ...	233670	72	-34.9015745,...
174...	633348	78	2014-12-12 18:51:41	NULL	0	2014-12-15 ...	232888	72	-34.9015745,...
177...	1271193	79	2014-12-14 19:52:04	NULL	0	2014-12-15 ...	56168	72	-34.9015745,...
177...	636069	82	2014-12-14 20:43:07	NULL	0	2014-12-15 ...	11935	72	-34.9016697,...

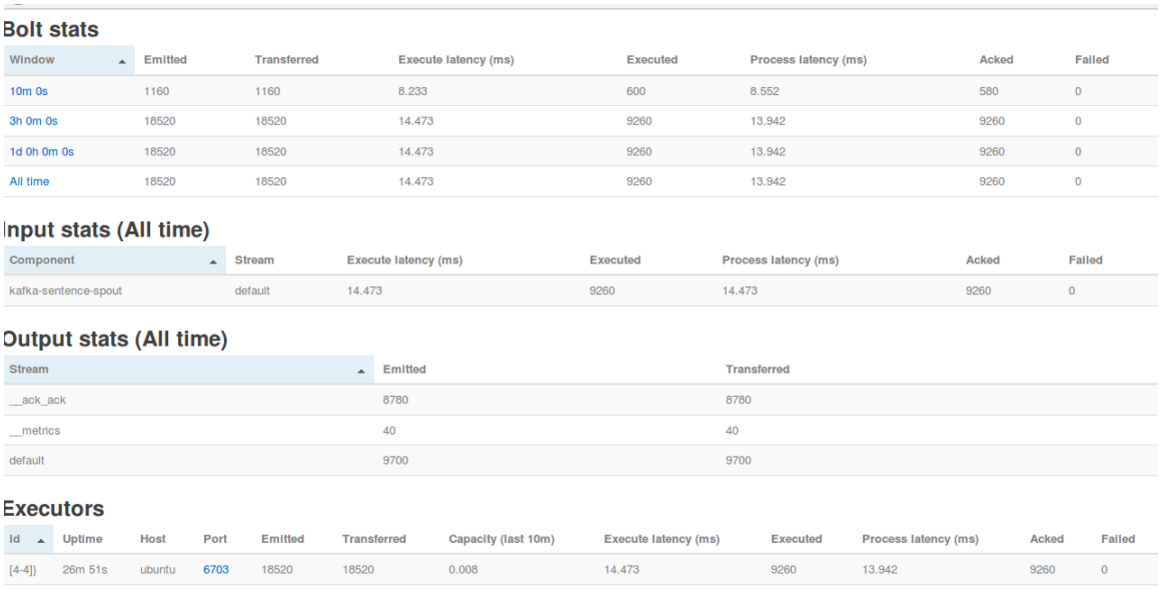
Result Grid | Form Editor

Figura 5.2: Comparativas de resultados entre el log de Storm y MySQL

Tomando en cuenta que se está simulando como en tiempo real a partir de la fecha 12/12/2014 17:02:13 (la fecha del primer evento), si deseamos ver la cantidad de personas suponiendo que la fecha actual es: 12/12/2014 23:50:28 nos devuelve el resultado de cuatro personas, comprobando con la consulta MySQL confirmamos que realmente son cuatro las personas que estuvieron en ese horario cumpliendo los filtros anteriormente mencionados.

Por otro lado observamos las métricas que genera la topología en la Figura 5.3. Podemos ver cuántas tuplas son emitidas y ejecutadas, por lo cual pudimos comprobar que se emiten la totalidad de las tuplas que son enviadas, sin generar pérdidas. Y por otro lado tenemos la latencia de ejecución, la cual claramente es una latencia

baja, una de las características que debe cumplir una solución para ser catalogada de Big Data.



Bolt stats

Window	Emitted	Transferred	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
10m 0s	1160	1160	8.233	600	8.552	580	0
3h 0m 0s	18520	18520	14.473	9260	13.942	9260	0
1d 0h 0m 0s	18520	18520	14.473	9260	13.942	9260	0
All time	18520	18520	14.473	9260	13.942	9260	0

Input stats (All time)

Component	Stream	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
kafka-sentence-spout	default	14.473	9260	14.473	9260	0

Output stats (All time)

Stream	Emitted	Transferred
__ack_ack	8780	8780
__metrics	40	40
default	9700	9700

Executors

Id	Uptime	Host	Port	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
[4-4])	26m 51s	ubuntu	6703	18520	18520	0.008	14.473	9260	13.942	9260	0

Figura 5.3: Storm UI para la topología de la solución

5.5. Trabajo a futuro

Una de las dificultades fue la obtención de los datos para que sea en tiempo real, por lo que se solucionó con una simulación. Para tener la solución con los datos reales en tiempo real se debería implementar un productor kafka en el servidor web de UnoWiFi para que envíe mensajes al broker donde Storm consumirá los datos. Desarrollada esa comunicación entre el broker y el servidor web se tendría un ambiente concreto ya que cada vez que un sensor envía información al servidor, este lo replicaría a un broker para que sea tratado en el momento por Storm.

Otra de las dificultades fue el hecho de contar con pocos datos a nivel Big Data. Esto se resolverá cuando se agrande el emprendimiento y cuente con mayor cantidad de sensores. En este caso se utilizaría otro sistema de almacenamiento como los que hemos mencionado en el capítulo 3 o contar con un servidor en la nube como es Amazon o Google que brindan herramientas para el manejo de grandes volúmenes de datos.

Otras de las posibles soluciones que se podrían llevar a cabo con estas tecnologías podrían ser los siguientes:

- A partir de un banco de datos con los perfiles de los usuarios que se conectan

a la red, se podría hacer llegar publicidad en tiempo real dirigida a cada uno de ellos de acuerdo a sexo, edad, gustos, etc.

- Visualizar en tiempo real el sistema operativo y marca de cada dispositivo conectado a la red.
- Realizar experimentos para calcular la velocidad promedio del tránsito en una determinada calle, teniendo como referencia varios sensores posicionados en puntos estratégicos de ciertas calles para así efectuar el cálculo de velocidad.
- Rastreo aproximado de dispositivos.

La empresa UnoWiFi puede hacer uso de esta solución para resolver su problema adaptándolo en un ambiente de tiempo real como mencionamos. Existen muchas soluciones a otros problemas que pueden ser implementados y generarían resultados positivos para la empresa y sus clientes.

5.6. Conclusiones

En este capítulo se ha podido desarrollar una solución aplicando dos herramientas en conjunto como lo son Apache Kafka y Apache Storm. Las mismas pertenecen a distintos componentes de la arquitectura de Big Data como son recolección y procesamiento de datos respectivamente mostrando que ambas pueden trabajar en conjunto para generar un sistema. Si bien los resultados para este problema no son del todo exactos, debido a las dificultades mencionadas en la sección 5.3.2, se pueden apreciar los diferentes casos de uso en los que puede ser utilizada esta tecnología.

Uno de los desafíos más costosos de la implementación fue la verificación y resolución de errores. Al trabajar con grandes volúmenes de datos a alta velocidad es complicado poder identificar por qué falla el sistema, por lo que hubo que disponer de nuevas técnicas de resolución de errores como lo son: la visualización de los archivos de log en donde se guardan todas las instrucciones generadas por el programa y enlentecer la simulación para identificar la falla.

Para la solución se tuvieron en cuenta diversas plataformas y lenguajes, con la dificultad que cada una presenta. Las tareas de instalación, configuración e integración de las herramientas no fueron triviales, teniendo que realizar varias pruebas para llegar a un ambiente estable. Luego de comprender los conceptos y disponer un ambiente estable de desarrollo, se pueden resolver una gran variedad de problemas relacionados al procesamiento en tiempo real.

El caso de estudio no solo presentó la dificultad de crear una lógica de procesamiento en tiempo real para obtener el resultado esperado sino todos los pasos previos que se realizaron para lograr una pureza de los datos, manipulándolos previamente para simular un ambiente en tiempo real, lo que sirvió para diseñar una solución que pueda ser expandida y adaptada a un sistema similar.

La resolución del problema nos ha permitido aplicar los conceptos adquiridos en un caso real. Hemos implementado una solución que puede ser adaptada a la realidad de la empresa. La innovación del Usuario Responsable puede ser aplicada en varios sectores como la seguridad, tránsito, economía, etc. para recopilar datos de interés, mejorar los servicios y/o productos y obtener resultados valiosos.

Se notó una cordial amabilidad de parte de Gustavo, ofreciéndonos ayuda e interesándose por nuestro trabajo, lo cual nos motivó en la búsqueda de otras posibles soluciones. De acuerdo al esperado crecimiento del volumen de datos, y manejo de tiempos de respuesta reducidos en cuanto a su procesamiento es esperable que la tecnología Big Data sea un nuevo desafío para la empresa.

Se pueden hacer muchas aplicaciones de utilidad en base a los datos que se obtienen y combinarlos con otros para enriquecerlos satisfaciendo las necesidades de sus clientes, ya que mejorarían los servicios y productos. Por ejemplo, una vez que se tenga definido un banco de perfiles de usuario, puede interesar detectar cierto perfil en un determinado lugar para luego mostrarle publicidad en tiempo real dirigida al mismo de acuerdo a su perfil.

Capítulo 6

Conclusiones y trabajo a futuro

Se presentan a continuación las conclusiones extraídas a partir del trabajo realizado y algunas sugerencias para posibles trabajos a futuro.

6.1. Conclusiones

Big Data es un tópico relativamente nuevo en el mundo de la informática y que en los últimos años ha logrado llamar la atención de una gran cantidad de investigadores y organizaciones. Este interés progresivo va de la mano con la realidad de que existe tanto, un constante aumento de la cantidad de información como la intención de obtener de ella un valor agregado.

Se identificaron dos grandes áreas en cuanto al procesamiento de la información se refiere, como lo son el procesamiento batch o en lotes y el procesamiento en tiempo real. Cada uno de ellos tiene objetivos específicos y herramientas hechas a medida para afrontarlos. Se presentaron arquitecturas que tienen como objetivo ofrecer mecanismos estandarizados para la integración de estas tecnologías y permitir la construcción de soluciones que permitan tanto el procesamiento batch y en tiempo real.

Existen distintas organizaciones que han puesto especial énfasis en la expansión global de los conceptos y tecnologías que se utilizan para la generación de soluciones a problemas relacionados a Big Data, tales como Oracle, IBM, el Proyecto Apache, etc. Ésta última constituye un pilar fundamental en el desarrollo de este tipo de soluciones, teniendo como bastión principal el proyecto Hadoop. Por su condición de software libre provee a desarrolladores de herramientas gratuitas y de constante evolución a partir del aporte de la comunidad mundial que nutre a este proyecto. Están disponibles numerosas opciones a la hora de seleccionar una herramienta dada, que dependerá exclusivamente del problema que se tenga.

Otro de los objetivos planteados fue realizar un estudio de las herramientas existentes, describiendo sus características principales. Existen diversos recursos para la implementación de trabajos dirigidos al procesamiento por lotes y en tiempo real. Hadoop proporciona instrumentos que les permiten a los desarrolladores disponer de recursos para afrontar ambos tipos de problemas. Con sus distribuciones integrales, como lo son Hortonworks y Cloudera, los programadores pueden usufructuar los artefactos de Hadoop de forma más amigable y sencilla, ya que disponer de un ambiente con varios de ellos instalados permite tener un manejo más cómodo de los mismos.

Hemos realizado varios casos prácticos utilizando las herramientas Apache Kafka, Apache Storm y la distribución Hortonworks, que se adaptan al enfoque de toma de decisiones en tiempo real. Se ha logrado una adecuada familiarización con dichas herramientas para luego abordar un caso de estudio basado en una situación real de la empresa UnoWiFi , el cual consiste en obtener el conteo de personas en cierto establecimiento en todo momento.

En base a las herramientas analizadas se ha implementado una solución al caso de estudio obteniendo resultados favorables y útiles en la práctica. Si bien la cantidad de datos brindados por la empresa UnoWiFi para el caso no era masiva para tratarse como problema de Big Data, hemos diseñado e implementado un sistema que, ejecutado en un ambiente con dichas herramientas se pueden procesar datos y obtener resultados de interés en tiempo real. Con un mayor desarrollo se podría construir un producto con posibilidades de competir en el mercado. El sistema implementado podría ser complementado con procesamiento por lotes (arquitectura lambda) y así generar una solución integral. Por ejemplo, un banco de perfiles de usuario se podría obtener a partir del procesamiento batch.

Pensamos que la solución obtenida le abre una puerta a la empresa para la introducción de tecnología Big Data, que de acuerdo a las expectativas de crecimiento que se tienen es muy factible que en un futuro sean aplicables.

Debemos señalar que surgieron ciertos inconvenientes en el transcurso del proyecto. Se destacan como problemas principales los relativos a la configuración e instalación de algunas herramientas no tan difundidas y que carecían de documentación atrasando nuestra planificación.

Por otro lado se destaca el aprendizaje y conocimiento adquirido. El trabajo permitió contextualizar y comprender el concepto de Big Data, junto a muchas de las aristas que componen esta área. El que sea un tema actual, que está adquiriendo cada vez más difusión, fue para nosotros muy atrayente y motivador. Consideramos muy positivo el uso que se le puede llegar a dar para mejorar áreas relativas a la calidad

de vida de las personas tales como salud, seguridad, previsión de desastres naturales, etc.

Podemos concluir que se logró cumplir con el objetivo general planteado y con los objetivos específicos como se menciona anteriormente, pues se realizó un relevamiento amplio del estado del arte de Big Data así como el análisis de herramientas realizando un caso práctico de una aplicación real para satisfacer los requerimientos del Usuario Responsable.

6.2. Trabajo a futuro

En esta sección se presentan sugerencias acerca de posibles trabajos futuros a realizar. Algunas de las proposiciones refieren a actividades generales y otras a los resultados obtenidos en el proyecto.

1. Continuar con el estudio del área, ya que es constante el crecimiento que presenta. Si bien no es el tema más popular hoy en día, es evidente que será uno de los más estudiados en los años que se avecinan.

2. Proponer trabajos para el desarrollo de aplicaciones utilizando las herramientas existentes, de forma de acercarse de mejor manera a los conceptos e ideas del área. Las soluciones de Big Data pueden ser aplicadas en diversas áreas (seguridad, economía, marketing, comercial, etc.) no solo para mejorar sino también para resolver problemas de los cuales no se puede encontrar una mejor solución.

3. Realizar un seguimiento de futuras versiones de las herramientas estudiadas, ya que dada la expansión del rubro es constante la evolución de las mismas, proporcionando así más oportunidades para el desarrollo de aplicaciones.

4. Hemos identificado un amplio abanico de posibilidades en cuanto al uso de técnicas de Big Data por parte de UnoWifi. Nos basamos en las proyecciones de crecimiento del volumen de datos que se manejará en el corto plazo, y en las prestaciones que se desean brindar a los clientes de la empresa. Se conjugará el tratamiento de ese gran volumen con la capacidad de brindar información de interés a los usuarios de la red en tiempo real. Una de las necesidades más importantes de la empresa UnoWiFi para satisfacer a sus clientes es la construcción de una base de datos que contenga los perfiles de los usuarios. De esta manera se podrán obtener estadísticas en tiempo real de varios aspectos (conocer el rango de edad, género, los gustos de cada persona para generar

marketing digital, mostrar publicidad según el público que se tiene, etc.). Otro de los desafíos que va a presentar la empresa es migrar su sistema actual a uno Big Data, manejando bases de datos que permitan operar con grandes volúmenes de información (bases NoSQL).

Bibliografía

[1] The Big Data opportunity. C. Yiu. *Policy exchange, London - ISBN: 978-1-907689-22-2*, 2012.

[2] Big Data: The next frontier for innovation, competition and opportunity. J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers. *McKinsey Global Institute, New York, NY - ISBN: 978-0-983179-69-6*, 2011.

[3] Undefined by data: A survey of Big Data definitions. J. S. Ward, A. Barker. University of St Andrews, UK. *CoRR - arXiv:1309.5821*, 2013.

[4] Critical questions for Big Data: Provocations for a cultural, technological, and scholarly phenomenon. D. Boyd, K. Crawford. *Journal Information, communication and society, Vol. 15, 662-679*, 2012.

[5] Application controlled demand paging for out of core visualization. M. Cox, D. Elisworth. NASA Ames Research Center - Moffett Field, CA. *Proceedings of the 8th conference on Visualization'97, 235-244*, 1997.

[6] Imagen de Información generada en Internet en un minuto. Ult. Acceso: 2014.06.10. http://www.elmundo.com.ve/noticias/tecnologia/internet/2014-sera-el-ano-del--big-data-.aspx.

[7] Información sobre Big Data - Sitio oficial de IBM. http://www-01.ibm.com/software/data/bigdata.

[8] The age of Big Data. S. Lohr. *New York Times, Vol. 11*, 2012.

[9] ¿Qué es Big Data?. Sitio oficial de IBM. Developerworks. Ricardo Barranco Fragoso. Ult. Acceso: 2014.06.02. http://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/.

[10] Internet será cuatro veces más grande en 2016. Sitio oficial Cisco. Ult. Acceso: 2014.05.30. http://www.cisco.com/web/ES/about/press/2012/2012-05-

30-internet-sera-cuatro-veces-mas-grande-en-2016--informe-vini-de-cisco.html.

[11] IBM insight - Sitio oficial. Ult. Acceso: 2014.11.22. http://www-01.ibm.com/software/events/insight.

[12] The Big Data - Driven Business: How to use Big Data to win customers, beat competitors, and boost profits. R. Glass, S. Callahan. *Wiley, Hoboken, New Jersey - ISBN: 978-1-118-88980-0*, November 2014.

[13] Netflix sees major success due to Big Data analytics programs. Ult. Acceso: 2015.03.04. http://www.attunity.com/learning/articles/netflix-sees-major-success-due-big-data-analytics-programs.

[14] Portal de datos abiertos de EUA. Ult. Acceso: 2015.03.03. http://www.data.gov/.

[15] Portal de datos abiertos de Uruguay. Ult. Acceso: 2015.03.03. http://datos.gub.uy/.

[16] Big Data a review. S. Sagiroglu, D. Sinanc. Gazi University. Department of Computer Engineering, Faculty of Engineering. Ankara, Turkey. *IEEE. CTS, 2013 International Conference on (San Diego, CA. 20-24 May), 42 - 47*, 2013.

[17] Understanding Big Data, analytics for enterprise class, Hadoop and streaming data. D. deRoos, C. Eaton, G. Lapis, P. Zikopoulos, T. Deutsch. *McGraw-Hill Osborne Media - ISBN: 978-0-071-79053-6*, 2011.

[18] NoSQL Database: New era of databases for big data analytics - classification, characteristics and comparison - A B M Moniruzzaman, S. A. Hossain - Department of Computer Science and Engineering Daffodil International University. *International Journal of Database Theory and Application - Vol. 6, No. 4*, 2013.

[19] Perspectives on the CAP theorem. S. Gilbert - National University of Singapore, N. A. Lynch - MIT. *IEEE, Journal Computer, 30-36*, 2012.

[20] MapReduce online. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein UC Berkeley - K. Elmeleegy, R. Sears Yahoo! Research. *NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation, Vol. 10, 21-22*, 2010.

[21] Big Data: Big opportunity in banking. The financial brand. Ult. Acceso: 2014.07.07. http://thefinancialbrand.com/26363/big-data-analytics-retail-banking-jm/.

[22] Discovery of extreme events-related communities in contrasting groups of physical system networks. Z. Chen, W. Hendrix, H. Guan, I. K. Tetteh, A. Choudhary, F. Semazzi, N. F. Samatova. *Springer. Journal Data Mining and Knowledge Discovery, 225–258*, 2013.

[23] Caso de uso smart steps. Telefonica. Ult. Acceso: 2014.07.08. http://dynamicinsights.telefonica.com/488/smart-steps.

[24] Telefónica lanza "Telefónica Dynamic Insights", una nueva unidad de negocio global de Big Data. Director Stephen Shurrock. 2012. http://saladeprensa.telefonica.es/.

[25] Geospatial Big Data: Challenges and opportunities. JG Lee, M. Kang. *Big Data Research. Vol. 52 No. 2, 74-81*, 2015.

[26] Amazon Kinesis - Sitio oficial. Ult. Acceso: 2014.08.14. http://aws.amazon.com/es/kinesis/.

[27] Cloud platform at Google I/O - new Big Data, mobile and monitoring products. Greg Demichillie. Google Developers. 2014.

[28] Google Big Query - Sitio oficial. Ult. Acceso: 2014.08.13. https://cloud.google.com/bigquery/.

[29] Datasift architecture: Realtime datamining at 120,000 tweets per second. Ult. Acceso: 2014.08.17. http://highscalability.com/blog/2011/11/29/datasift-architecture-realtime-datamining-at-120000-tweets-p.html.

[30] Datasift - Sitio oficial. Ult. Acceso: 2014.08.17. http://www.datasift.com/.

[31] Infraestructura de Datasift - Sitio oficial de Datasift. Ult. Acceso: 2015.03.04. http://datasift.com/platform/infrastructure/.

[32] Lambda architecture for real time Big Data analytic. Z. Hasani, M. K. Popovska, G. Velinov. *ICT Innovations 2014 Web Proceedings, 133-143*, 2014.

[33] Imagen de arquitectura lambda. Sitio oficial Lambda. Ult. Acceso: 2014.10.09. http://lambda-architecture.net/.

[34] Big Data analytics - From strategic planning to enterprise integration with tools, techniques, NoSQL, and graph. D. Loshin. *Elsevier, Morgan Kaufmann, San Francisco, CA - ISBN: 978-0-12-417319-4*, 2013.

[35] Use cases for Hadoop. Cloud Partner STM. Ult. Acceso: 2014.05.03. http://www.cloudpartnerstm.com/wp-content/uploads/2012/09/Use-Cases-for-Hadoop.pdf.

[36] Using Big Data for intelligent businesses. C. Bucur. *Proceedings of the Scientific Conference AFASES. May2015, Vol. 2, 605-612.*, 2015.

[37] How Hadoop can help keep your money in the bank. Derrick Harris. Ult. Acceso: 2014.05.03. http://gigaom.com/2012/03/08/how-hadoop-can-help-keep-your-money-in-the-bank.

[38] Top ten Big Data security and privacy challenges. CSA (Clouad Securty Allian-ce). 2012.

[39] Apache Hadoop - Who. Ult. Acceso: 2014.04.25. http://hadoop.apache.org/who.html.

[40] Usuarios de Apache Hadoop. Ult. Acceso: 2014.04.25. http://wiki.apache.org/hadoop/PoweredBy.

[41] La historia de Hadoop. Ult. Acceso: 2014.04.29. https://gigaom.com/2013/03/04/the-history-of-hadoop-from-4-nodes-to-the-future-of-data/.

[42] The Google File System. S. Ghemawat, H. Gobioff, S.T. Leung. Lake George, NY. *ACM SIGOPS operating systems review, Vol. 37, 29-43*, 2003.

[43] MapReduce: simplified data processing on large clusters. J. Dean and S. Ghemawat. *OSDI'04: Proceedings of the 6th conerence on Symposium, Vol 6, 10-10*, 2004.

[44] Apache Ambari - Sitio oficial Hortonworks. Ult. Acceso: 2014.04.29. http://hortonworks.com/hadoop/ambari/.

[45] Apache Zookepeer - Sitio oficial. Ult. Acceso: 2014.09.29. http://zookeeper.apache.org/.

[46] Zookeeper - A reliable, scalable distributed coordination system. Ult. Acceso: 2014.09.29. http://highscalability.com/blog/2008/7/15/zookeeper-a-reliable-scalable-distributed-coordination-syste.html.

[47] Arquitectura Apache Chukwa - Sitio oficial. Ult. Acceso: 2014.11.21. https://chukwa.apache.org/docs/r0.4.0/design.html.

[48] Arquitectura de Apache Flume - Sitio oficial. Ult. Acceso: 2014.09.29. https://flume.apache.org/FlumeUserGuide.html.

[49] Apache HBase - Sitio oficial. Ult. Acceso: 2014.11.23. http://hbase.apache.org/.

[50] Bigtable: A distributed storage system for structured data. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber. Seattle, WA. *Journal ACM Transactions on Computer Systems (TOCS), Vol. 26, No. 2, 4-4*, 2008.

[51] HBase: The definitive guide - 2nd edition. M. Stack. *O'Reilly Media, Sebastopol, CA - ISBN: 1-4493-9610-0*, 2011.

[52] Apache Pig - Sitio oficial. Ult. Acceso: 2014.11.23. http://pig.apache.org/.

[53] Apache Mahout - Sitio oficial. Ult. Acceso: 2014.11.24. http://mahout.apache.org/.

[54] Apache Oozie - Sitio oficial. Ult. Acceso: 2014.11.22. http://oozie.apache.org/.

[55] Hortonworks - Sitio oficial. Ult. Acceso: 2014.04.29. http://hortonworks.com/.

[56] Información de Cloudera - Sitio oficial. Ult. Acceso: 2014.11.12. http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/introduction.html.

[57] Arquitectura de Cloudera - Sitio oficial. Ult. Acceso: 2014.04.29. http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_intro.html.

[58] Arquitectura de Cloudera. 2015. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html.

[59] Apache Kafka - Sitio oficial. Ult. Acceso: 2014.04.29. http://kafka.apache.org/.

[60] Anatomía de un topic - Sitio oficial Apache Kafka. Ult. Acceso: 2015.03.04. http://kafka.apache.org/documentation.html.

[61] History of Apache Storm and lessons learned. Blog de Nathan Marz. Ult. Acceso: 2015.03.04. http://nathanmarz.com/blog/history-of-apache-storm-and-lessons-learned.html.

[62] Apache Storm - Sitio de Hortonworks. Ult. Acceso: 2014.04.20. http://hortonworks.com/hadoop/storm/.

[63] Apache Storm - Sitio oficial. Ult. Acceso: 2014.04.20. https://storm.apache.org.

[64] Storm Twitter. Twitter Inc. Univesity of Wisconsin Madison. A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mitta, D. Ryaboy. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 147-156*, 2014.

[65] Apache S4 - Sitio oficial. Ult. Acceso: 2014.01.05. http://incubator.apache.org/s4/.

[66] S4: Distributed stream computing platform. L. Newmeyer, B. Robbins, A. Nair, A. kesari : Yahoo! Labs. *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on, 170-177*, 2010.

[67] S4 vs storm. University of Glasgow. Richard McCreadie. 2013.

[68] Descarga de Hortonworks - Sitio oficial. Ult. Acceso: 2014.01.12. http://hortonworks.com/products/hortonworks-sandbox/#install.

[69] Descarga de VMWare - Sitio oficial. Ult. Acceso: 2014.01.12. https://my.vmware.com/web/vmware/downloads.

[70] Kafka: a distributed messaging system for log processing. J. Kreps, N. Narkhed, J. Rao: LinkedIn Corp. *Proceedings of the NetDB, 1-7*, 2011.

[71] ActiveMQ - Sitio oficial. Ult. Acceso: 2014.06.10. http://activemq.apache.org/.

[72] RabbitMQ - Sitio oficial. Ult. Acceso: 2014.06.10. http://www.rabbitmq.com/.

[73] Código proyecto storm-starter - Repositorio Github. Ult. Acceso: 2015.03.04. 2015. https://github.com/apache/storm/tree/master/examples/storm-starter.