Universidad Carlos III de Madrid
Escuela Politécnica Superior
Grado en Ingeniería en Tecnologías de Telecomunicación

TRABAJO FIN DE GRADO

Sistema *big data* para el análisis de rutas de taxis en NYC

Autor: Montserrat Murillo González

Tutor: Pablo Basanta Val

Leganés, Marzo de 2016

Universidad
Carlos III de Madrid

Título: Sistema *big data* para el análisis de rutas de taxis en NYC

Autor: Montserrat Murillo González

Director: Pablo Basanta Val

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de Marzo de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de ___.

SECRETARIO VOCAL PRESIDENTE

Agradecimientos

En primer lugar dar las gracias a mi tutor Pablo por ayudarme a realizar este proyecto y por darme la oportunidad de aprender un poco más del mundo del big data.

Por encima de todo, dar las gracias a mi familia, a mis padres y mi hermano con cuya ayuda, esfuerzo y apoyo me han acompañado en todos y cada uno de los logros conseguidos. Gracias a mi hermano, por sus consejos infinitos, por su paciencia y por guiarme en todo momento cuando más lo necesito. A mi madre por ser experta en quitarle hierro al asunto cuando más agobiada y saturada estoy y por los esfuerzos hechos para que pudiera conseguir lo que me he propuesto. Y a mi padre por recordarme siempre que las carreras de fondo tienen fin y que al final todo esfuerzo tiene recompensa.

Por último a todos aquellos que me han ayudado y animado y sobre todo han confiado en mí.

Gracias.

Resumen

Con el incipiente aumento de los datos informáticos que se generan en la actualidad surge el término *big data.* Las herramientas convencionales utilizadas hasta el momento no son capaces de soportar los grandes volúmenes de datos, por lo que se tiene que recurrir a las nuevas tecnologías del momento.

El uso de las tecnologías de información más avanzadas permite la búsqueda de respuestas en el menor tiempo posible, lo que conlleva la necesidad del tratado de los datos de forma rápida. Muchas organizaciones ya recurren a estas herramientas, ya que a parte de la velocidad en la respuesta les permite analizar y obtener conocimiento a partir de todo el conjunto de datos que poseen. Además, tienen la posibilidad de realizar la toma de decisiones en tiempo real y mejorando así sus estrategias de negocio.

La aplicación del *big data* se realiza en multitud de sectores, y en este proyecto el sector central es el de tráfico. Se ha hecho un estudio del diseño de un posible sistema *big data* que ofrece al usuario la capacidad de cargar, procesar y visualizar los datos de entrada al sistema con las tecnologías *Hadoop* y *Hive*.

En esta memoria se comienza dando una visión general de la motivación del problema al que se pretende dar solución, seguida de unos capítulos introductorios a la tecnología empleada en el desarrollo. Posteriormente, se expone el diseño y los pasos seguidos en la implementación del sistema continuando con las pruebas de rendimiento. Finalmente, se concluye con las conclusiones de la realización y la planificación y presupuesto del proyecto.

Palabras clave: datos, big data, tecnología de información, volumen, velocidad, Hadoop, Hive,

Abstract

In the recent years, increasing computer data has given rise to the *big data* term. Conventional tools used so far are not able to support large volumes of data, so you have to resort to new technologies of the moment.

Start using the most advanced information technologies allows searching for answers in the shortest time possible, which means that data processing must be fast. Many organizations already use these tools, which offer them the response speed and it allows them to analyze and gain knowledge from the entire set of data held. Additionally, they have the ability to make decisions in real time and thus improving their business strategies.

The application of *big data* is done in many sectors and specifically this project is included in the traffic sector. We performed a study of the design of a possible big data system that gives users the ability to upload, process and display the data input to the system with *Hadoop* and *Hive* technologies.

This document is divided into several parts. Firstly, the project's motivation and context are explained and continues with some chapters of introduction to the technology to be used in the project. After that, the design and implementation of big data system are described. In addition, the performance testing is also explained. Finally, it ends with conclusions, stages of development and the project budget.

Keywords: data, big data, information technologies, volume,

velocity, Hadoop, Hive,

Contenido

Índice de figuras

Índice de tablas

Bloque I

Introducción

Capítulo 1

1. Introducción y objetivos

1.1. Contexto y motivación

Como consecuencia del avance de las tecnologías de información surgen nuevas necesidades que las herramientas tradicionales son incapaces de suplir [1]. Esto se debe al aumento del volumen de datos que hace del procesado una etapa costosa a la hora de buscar respuestas.

Por esta razón, las organizaciones se han tenido que enfrentar a nuevos desafíos para la obtención de conocimiento o información que no puede conseguirse a priori a partir del gran conjunto de datos generados en la actualidad. El uso de las nuevas tecnologías les ayuda a hacer un análisis, descubrir y poder entender más allá de lo que las herramientas convencionales les ofrece sobre sus datos, y así aumentar su competencia.

Este proyecto, se enmarca en el sector de transporte ya que se utilizan datos generados por taxis en Nueva York aportados por el concurso DEBS 2015 Grand Challenge [2]. Se busca el procesado de estos datos con tecnologías que los soporten, es decir, tecnologías *big data* para extraer información sobre las rutas más frecuentes por horas de cada día, en los medios días y por días completos. Esta información ayudaría a entender el tráfico generado y a la toma de decisiones estratégicas en el sector.

Para la obtención de resultados, un usuario debe ser capaz de interactuar con el sistema *big data*. En primer lugar, una vez extraídos los datos de la fuente de origen, tienen que ser cargados en el sistema para ser tratados posteriormente. El procesamiento tiene como finalidad tanto la recolección de datos válidos, así como el reformateo de los mismos si fuera necesario. Por lo que se realiza un proceso de limpieza dónde se eliminan aquellos registros considerados erróneos, se mantienen los campos que van a ser utilizados y se transforman en el caso de que el sistema destino lo requiera. Por último, se realiza la visualización de los resultados, que en el caso de este proyecto serán rutas de vehículos sobre un mapa.

El sistema *big data* se desarrollará con la tecnología *Apache Hadoop* [3] y *Apache Hive* [4] utilizando las ventajas ofrecidas por los sistemas distribuidos y la abstracción de la programación de técnicas de procesamiento para grandes volúmenes de datos. Sin embargo, el diseño del sistema se adapta al desarrollo con otras tecnologías y a la modificación de la configuración del entorno de trabajo aportando de esta forma flexibilidad al sistema.

A continuación se ilustra el esquema general del sistema desarrollado en el proyecto (véase Figura 1), dando una visión global de las funcionalidades y las tecnologías y herramientas empleadas. Como se ha comentado anteriormente el usuario tiene la capacidad de cargar los datos en el sistema, en concreto, en el sistema de ficheros distribuido de *Hadoop* conocido como HDFS. Estos datos pueden provenir de diferentes fuentes de origen ocasionando la no homogeneidad de los datos en cuanto a formato y estructura. Sin embargo, en el caso del proyecto desarrollado los datos se obtienen del mismo origen. Después gracias al *framework Apache Hive* se procede a la limpieza del dato, es decir, se descartan los registros no válidos estableciendo las reglas de eliminación previamente y se crean algunos campos nuevos a partir de los ya existentes. Los campos no sufrirán ningún proceso de reformateo, ya que no hay ningún sistema de destino que lo precise. Como último paso, los datos obtenidos en la salida del procesado serán visualizados en un mapa con la herramienta de código abierto QGIS.

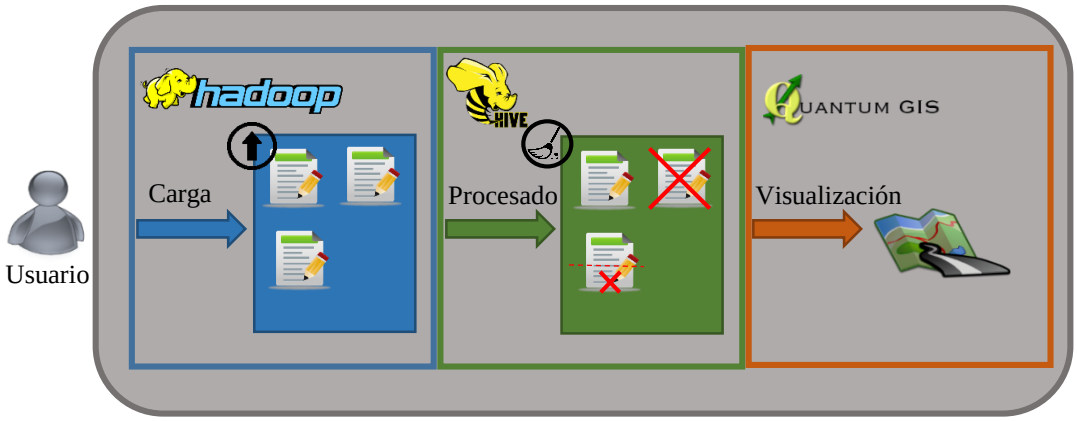


Figura 1. Esquema general del sistema big data desarrollado.

1.2. Objetivos

Este proyecto consiste en el desarrollo de un sistema *big data*. Los objetivos en los cuales se basan las fases de desarrollo son los siguientes:

1. Estudio del contexto y la motivación que llevan a la realización del proyecto.
2. Estudio de las tecnologías que van a ser utilizadas en el desarrollo del sistema.
3. Diseño de un sistema *big data* que se adapte a las necesidades del proyecto.
4. Implementación del sistema en base al diseño del mismo.
5. Visualización de los resultados del procesamiento de los datos.
6. Realización de pruebas de rendimiento del sistema en diferentes entornos.
7. Extracción de conclusiones del desarrollo del proyecto.
8. Planteamiento de líneas futuras ofreciendo continuidad al proyecto.

1.3. Estructura del documento

Para poder facilitar la estructura de la memoria se detalla a continuación una estructura de la misma:

- Bloque I: Introducción.

  - Capítulo 1: Introducción y objetivos.
    En este primer capítulo se exponen las motivaciones del proyecto y se establecen los objetivos del mismo recogiendo la idea general del trabajo que se va a realizar. Además, se enuncia la estructura de la memoria.

- Bloque II: Estado del arte.

  - Capítulo 2: Estado del arte.
    En el capítulo 2, se detalla las tecnologías utilizadas en el desarrollo del proyecto las cuales deben ser conocidas por el lector previamente a la exposición de la solución técnica.

- Bloque III: Trabajo realizado.

  - Capítulo 3: Diseño del sistema.
    Este capítulo se describe el diseño del sistema *big data*, así como los componentes que lo forman y la interacción entre ellos.
  - Capítulo 4: Implementación.
    El capítulo 4 recoge la descripción de los pasos que se han seguido en el desarrollo de la implementación del sistema diseñado, desde la carga inicial de datos hasta la visualización de los resultados obtenidos en el procesado.

  - Capítulo 5: Pruebas y resultados.
    En este capítulo se explican las pruebas que se han realizado para conocer los límites del entorno y los tiempos de ejecución alcanzados.

- Bloque IV: Conclusiones y futuras líneas de trabajo.

  - Capítulo 6: Conclusiones y futuras líneas de trabajo.
    En el capítulo 6, se muestran el análisis de la consecución de los objetivos y las posibles líneas futuras de trabajo a realizar.

- Bloque V: Planificación y presupuesto.

  - Capítulo 7: Planificación.
    El capítulo 7, recoge las fases de desarrollo del proyecto y su diagrama de Gantt correspondiente.

  - Capítulo 8: Presupuesto.

En este capítulo se exponen los recursos empleados y el presupuesto para la realización del proyecto.

- Bloque VI: Anexos.

    - Anexo A: Cuadrícula para la visualización de rutas.
    - Anexo B: Gráficas de tiempos de ejecución.
    - Anexo C: Extended abstract.
    - Anexo D: Normativa y marco regulador.
- Referencias

Bloque II

Estado del arte

Capítulo 2

2. Estado del arte

2.1. Big data

2.1.1. Introducción

Es un hecho que en la actualidad la generación de grandes cantidades de información digital aumenta de forma vertiginosa. La empresa IBM [5], una de las empresa con gran fuerza en el sector de las telecomunicaciones y en consultoría afirma que la cantidad de datos generados al día son 2,5 trillones de bytes, lo que se traduce en que el 90% de los datos existentes en el mundo hoy en día se han creados durante los dos últimos años [6].

El almacenamiento de datos no es una práctica nueva, ni mucho menos de la actualidad. El ser humano lleva haciéndolo desde tiempo inmemorables. Ya esculpían nuestros antepasados en las piedras y pintaban en paredes para llevar un recuento de los suministros que les quedaban para sobrevivir. Por ejemplo, el ábaco fue una herramienta de análisis y que se utilizaba en el cálculo cuando la mente humana precisaba de ayuda para ello. Otro paradigma, fueron las bibliotecas, que son un claro ejemplo de almacenaje de información [7].

De todo esto, surge el término *big data* (o macrodatos) como conjunto de herramientas capaces de procesar y analizar los grandes volúmenes de datos almacenados tanto estructurados como no estructurados o semiestructurados que no pueden ser tratados de forma convencional con las herramientas de software tradicionales [8]. Además, se clasifica como la tendencia o moda en el avance de la tecnología que implica un enfoque nuevo en la toma de decisiones.

2.1.2. Las 5V´s

El *big data* se define con las 5V´s (véase Figura 2). Aunque inicialmente fueron 3, las 5V´s de la actualidad son las siguientes [9]:

- Volumen: Es la característica más evidente por lo que el propio nombre *big data* indica. Este concepto no puede ser establecido en cantidades fijas de bytes tales que permanezcan invariante en el tiempo, ya que el avance tecnológico cada vez permite el procesamiento de una cantidad de datos cada vez mayor. Aun así, hoy en día ese límite determinado por la capacidad de las herramientas informáticas y se habla de Pentabytes (10^{15}) [8].
- Velocidad: Es un reto que los datos sean tratados en el menor tiempo posible. Las tecnologías tradicionales no pueden hacerlo de forma inmediata, por tanto el *big data* es clave para la toma de decisiones, ya que permite dar respuesta de forma rápida. Incluso el análisis de los datos se realiza en tiempo real, cuando las necesidades lo requieran.
- Variedad: Los datos pueden provenir de distintas fuentes de origen. De tal forma que no hay homogeneidad dentro del conjunto de éstos con la existencia de datos estructurados cuya procedencia son las bases de datos relacionales, datos semiestructurados como son los documentos XML o los datos almacenados en bases de datos NoSQL y no estructurados tales como los datos de las redes sociales, e-mail o imágenes por ejemplo [10]. Por esta razón, uno de los primeros pasos en su tratado es dar uniformidad a toda la información [7].
- Veracidad: Otro factor importante es garantizar la calidad del dato, porque determinados tipos pueden ser intrínsecamente inciertos. Hay veces que ni la limpieza de los datos puede detectar la inexactitud por lo que es fundamental ser previsor y hacer un plan de acción anticipado [11].
- Valor: Negocio es quien percibe el resultado del *big data*, y quien debe saber qué datos deben analizarse para conseguir los objetivos y cumplir con las expectativas propuestas. El fin de todo es aportar beneficios para las personas, ya sean empresas, gobierno o la sociedad [7].

Figura 2. Las 5 V's del big data.

2.1.3. Fuentes de datos

Ya se ha mencionado anteriormente que las fuentes de origen de los datos pueden ser muy variadas, lo que provoca la no homogeneidad y caracteriza al *big data*. Es difícil ser consciente de la cantidad de información digital que generamos en nuestra vida cotidiana ya que vivimos en un mundo en el que la tecnología avanza a un ritmo desenfrenado. Las oportunidades, beneficios y ventajas que nos proporciona el uso de dichas tecnologías hacen que no reparamos en la huella digital que dejamos en el día a día con el uso de ellas [12].

La cantidad de datos generada tiende a asociarse desde el desconocimiento a todo movimiento relacionado con nuestros *smartphones* u ordenadores. Pero estos datos van más allá de los dispositivos móviles y del uso de Internet. Por ejemplo, cuando usamos la tarjeta bancaria para pagar en centros comerciales. También cuando hacemos uso del transporte como coger un vuelo, pagos en peajes o incluso cuando mandamos un paquete por mensajería. Cuando llamamos por teléfono a una empresa y dicen que nuestra conversación va a ser grabada por seguridad. Y por supuesto, todos los datos de las redes sociales y compras online. Estos son algunos de los ejemplos de generación de datos entre muchos otros.

Como se puede ver, son actividades cotidianas y que cada vez ponemos más en práctica. Además, no sólo el ser humano genera datos, sino también la comunicación M2M [13] (machine to machine o máquina a máquina). Los sensores son un ejemplo de esta comunicación y contribuye al aumento de información. Por ejemplo pueden instalarse en medios de transporte para recoger la ruta seguida, o en carreteras para determinar el número de vehículos, pueden medir precipitaciones o consumo eléctrico entre otros, y son utilizados por las compañías de tal forma que puedan explotarse y obtener información extra a partir de ellos.

2.1.4. Propósitos y casos de uso

Big data, con todo lo comentado hasta el momento aparece como un cambio disruptivo. Se pretende proporcionar una nueva realidad dirigida a las organizaciones para dar otro enfoque al mercado y generar así nuevas oportunidades en su capitalización [14].

El objetivo más inmediato es la búsqueda de conocimiento a partir de los datos para ayudar a la toma correcta de decisiones incluso en tiempo real, pero también es una oportunidad de negocio. El uso del *big data* en las empresas cada vez abunda más, ya que debido a su utilización se tiene una mayor percepción de las necesidades de los clientes y permite mejorar las prestaciones de los productos y servicios ofrecidos [8]. Incluso, se obtienen beneficios gracias a que se puede discernir el sentimiento interno de la empresa pudiendo actuar dentro de la organización.

Muchas de las tiendas de ropa utilizan *big data* para obtener información de las consultas en sus páginas web sobre los productos y analizan tendencias en las redes sociales para poder optimizar así el stock de sus almacenes o hacer campañas dirigidas de marketing. Además, establecen precios dinámicos sin necesidad de hacer un estudio exhaustivo de los competidores o de las acciones de los clientes, sólo con ayuda de las nuevas herramientas más potentes para el análisis de la información [7].

El grupo Inditex [15], es un ejemplo de distribuidor de moda se beneficia de las ventajas que les proporciona el uso del *big data*. Tiene un *Data Center* y una sala de operaciones desde donde extraen información en tiempo real de las tendencias. Esto les permite hacer colecciones adaptadas al consumidor y así diseñarlo y ponerlo a su disposición en un tiempo récord [16].

En relación a las entidades financieras, se plantea un reto importante y una gran oportunidad para poder acabar con las brechas de seguridad, los delitos financieros y el fraude [7]. Además, como en el caso de la industria textil, buscan conocer las necesidades de sus clientes y adaptar los servicios para satisfacerlas.

Uno de los proyectos de BBVA [17] es conocer a través de toda la información que poseen de sus clientes el riesgo que tiene la concesión de un crédito para pequeños y medianos comerciantes. También quieren compartir sus datos para brindar la oportunidad de explotarlos por otras entidades y así sacarlos mayor partido [18].

Otro sector que se beneficia de las nuevas tecnologías es el de las telecomunicaciones, con ayuda de los dispositivos móviles y de sus redes. Extraen información acerca de la experiencia de los usuarios a través del tráfico de voz y datos. De esta forma, pueden ofrecer tarifas personalizadas y hacerse fuerte en el mercado.

Telefónica, en concreto el departamento de I+D de Barcelona, ha participado en un proyecto para el desarrollo de un algoritmo predictivo de delitos cometidos en una zona. Se ha conseguido predecir en un 70% los crímenes producidos en la ciudad de Londres [19]. Por tanto, las fuerzas de seguridad también se unen al uso del *big data* ya que pueden utilizarlo para la detección de crímenes o incluso combatir con la lucha contra el terrorismo [7].

Tampoco se queda atrás la sanidad, con el cruce de información de historiales clínicos, entorno o incluso el clima pueden obtener modelos predictivos y mejor en calidad tanto el servicio en centros como la previsión y detección de enfermedades. Otro ejemplo, son las Smart cities (o ciudades inteligentes), que permite la mejora en la gestión de recursos en la ciudad gracias a la información proporcionada por los sensores.

Estos son algunos casos de uso del *big data* entre cientos que aprovechan las capacidades que ofrece. El uso de las nuevas tecnologías se adapta a las necesidades individuales, sólo hay que entender el negocio y tener claras las expectativas que se van a manejar.

2.1.5. Minería de datos

El *datamining* (o minería de datos) [20] es el proceso por el cual mediante un conjunto de técnicas y tecnologías se extrae información adicional de grandes conjuntos de datos. De esta forma se pueden extraer patrones repetitivos, tendencias o reglas que las tecnologías tradicionales son pueden detectar [21].

La minería de datos surge para ayudar a explicar el comportamiento de los datos dentro de un contexto determinado y buscar información oculta. Esto se hace mediante el uso de prácticas estadísticas, algoritmos de búsqueda y redes neuronales para la búsqueda de correlaciones entre los datos [22].

A continuación se van a comentar los escenarios de aplicación de la *minería de datos* y su metodología.

Escenarios de aplicación

Los patrones, tendencias y reglas se definen como un modelo de *datamining* y los escenarios donde se aplican pueden ser [23]:

- Predictivos: Para predecir acontecimientos.
    - Basados en clasificación: Separación de individuos o eventos en grupos relacionados para poder analizar y predecir afinidades de cada conjunto. Por ejemplo, si un cliente de una aseguradora dará un parte o no en un periodo de tiempo de acuerdo con su descripción (sexo, edad y/o domicilio).
    - Basados en regresiones: Cálculos destinados a la predicción. Por ejemplo, tiempo de inactividad de un servidor con los datos que se tienen de su inactividad en un tiempo pasado.

- Descriptivos: Para especificar características.
    - Basados en agrupación: Clasificación de individuos en base a sus características. Por ejemplo, el tipo de dolencia que puede tener un paciente por los antecedentes clínicos comunes con otros pacientes.
    - Basado en reglas de asociación: Establecer un vínculo entre determinados datos o campos. Por ejemplo, conocer que un cliente que compra un determinado producto además comprará otro.
    - Basado en secuenciación: Intentar predecir el valor de una variable en función del tiempo. Por ejemplo, el estudio de la demanda energética.

Metodología

La generación del modelo de minería de datos es un proceso cíclico (véase Figura 3) y comprende desde la formulación de la pregunta sobre los datos para la búsqueda de repuestas hasta la implementación del modelo en un entorno de trabajo. Después de la exploración de los datos puede resultar que se descubra que son ineficientes y por tanto

no sirven para la creación del modelo o se puede continuar con la generación de varios modelos y descubrir así que no responden al problema planteado por lo que debe volverse a definir el problema. También puede suceder que surjan nuevos datos a incorporar en el modelo por lo que puede haber varias iteraciones hasta crear el modelo definitivo [21].

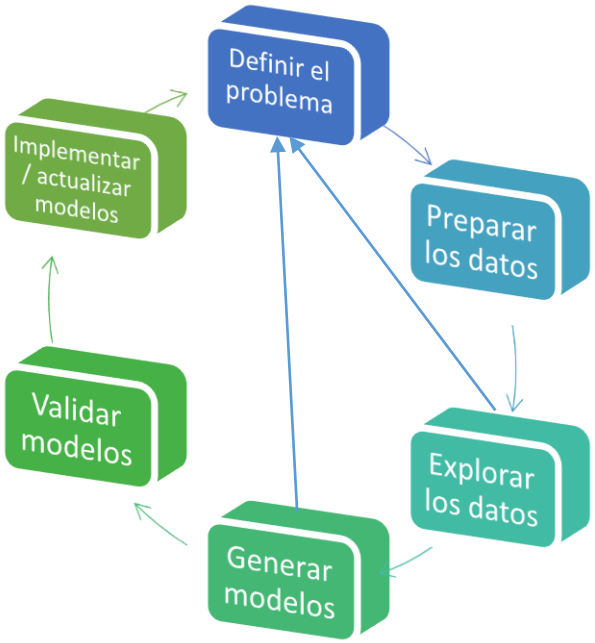


Figura 3. Proceso de generación del modelo de minería de datos.

Las fases de desarrollo para la generación final del modelo de *minería de datos* son las siguientes [21]:

1. Definir el problema: Consiste en el estudio del uso de los datos para dar respuesta al problema que se plantea. Para ello se debe analizar los requisitos empresariales y definir el ámbito del problema, las métricas de evaluación del modelo y los objetivos del proyecto.
2. Preparar los datos: Se deben entender los datos y prepararlos para la exploración. Los datos puede provenir de distintos orígenes por lo que hay que prepararlos para determinar con qué información nos vamos a quedar y que registros deben ser descartados para el análisis.
3. Explorar los datos: Para estudiar los datos normalmente se calculan máximos y mínimos, medias o desviaciones y se examina la distribución de los datos. Una vez explorados, se pueden desestimar los datos por ser insuficientes o defectuosos o se puede continuar con la generación del modelo y descubrir así comportamientos típicos.
4. Generar modelos: En la creación del modelo se debe determinar que campos de los datos son utilizados en la entrada, el atributo a predecir, y los parámetros que definen el algoritmo utilizado para el procesamiento de los datos. El procesamiento de un algoritmo es conocido como *entrenamiento*, ya que se

aplican cálculos matemáticos a los datos de entrada de forma iterativa para la extracción de patrones. Estos patrones dependen de los datos de entrenamiento y del algoritmo utilizado.

5. Validar modelos: La validación del modelo consiste en probar su correcto funcionamiento antes de ponerlo en producción y elegir la configuración con la que se obtienen resultados que más se ajustan a la definición del problema.
6. Implementar y actualizar modelos: Consiste en poner en producción el modelo elegido para crear predicciones y poder establecer toma de decisiones en función de los resultados, se pueden consultar los resultados y recuperar estadísticas. También pueden actualizarse los modelos tras una revisión o análisis y esto supone iniciar el proceso de nuevo.

Tras la breve explicación de la minería de datos es fácil vincularlo con *big data*, ya que ayuda a la extracción de conocimiento no inmediato después del análisis de los datos con técnicas matemáticas [24].

2.2. Computación distribuida

2.2.1. Introducción

La computación distribuida [25] es un modelo de computación en paralelo destinado a resolver problemas de computación masiva utilizando un conjunto de computadoras separadas físicamente pero conectadas a través de una red de comunicaciones. Sin embargo, de cara al usuario se comportan como una única entidad y puede acceder a los recursos remotos de la misma forma que lo hace para los recursos locales. Cada máquina se encarga de un fragmento de datos que forman parte del total, por lo que se dividen el trabajo mediante tareas individuales y una vez finalizadas se juntan los datos para obtener el resultado final [26].

Las ventajas que proporciona el uso de sistemas distribuidos son las comentadas a continuación [27]:

- Compartir recursos: Estos sistemas tienen la posibilidad de poder compartir los recursos hardware y software de las computadoras.
- Apertura: Se realiza el diseño sobre protocolos estándares de tal forma que se pueden combinar el equipamiento y software de distintos fabricantes.
- Concurrencia: Pueden existir varios procesos simultáneos en diferentes máquinas de la red.
- Escalabilidad: Tienen la posibilidad de aumentar la capacidad del sistema mientras se permita, añadiendo así recursos nuevos que cubren mayor demandas del sistema.
- Tolerancia a fallos: Se pueden utilizar otros sistemas cuando uno de ellos falla.

La computación distribuida a pesar de contar con ventajas importantes también presenta desventajas, que son las siguientes [27]:

- Complejidad: Tienen una mayor complejidad en la gestión que los sistemas centralizados.
- Seguridad: Dado que las conexiones se hacen de forma remota pueden surgir problemas a la hora de controlar el acceso a las distintas máquinas y pueden existir escuchas indeseadas en el intercambio de información.
- Manejabilidad: Las máquinas que forman el sistema pueden ser de diferentes tipos y cada una poseer un sistema operativo distinto.
- Impredecibilidad: La respuesta que ofrecen estos sistemas pueden variar de acuerdo a las circunstancias.

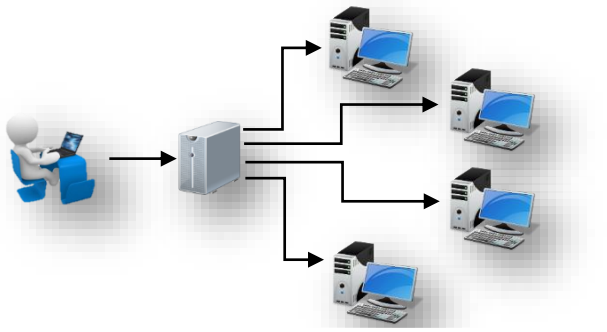


Figura 4. Sistema distribuido.

2.2.2. Cluster

Un método para crear sistemas distribuidos es el *cluster*. Se trata de un conjunto de máquinas con componentes hardware en común vistas en conjunto como una única computadora. Este tipo de sistema distribuido se ha utilizado por ejemplo para aplicaciones de supercómputo, en servidores web y comercio electrónico o en bases de datos de alto rendimiento [28].

En relación a los beneficios obtenidos por la tecnología cluster cabe destacar la velocidad de procesamiento que ofrece un *cluster* de alto rendimiento, así como la disminución del tiempo de respuesta que caracteriza a los *clusters* de balanceo de carga permitiendo mayor número de transacciones realizadas. Además se aumenta la confiabilidad y robustez en los *clusters* de alta disponibilidad [28].

La clasificación de los tipos de *cluster* se realiza en función de los servicios que ofrece y de sus características. Esa clasificación es [29]:

- Alto rendimiento (HPC- High Performance Clusters): Para tareas que requieren gran capacidad computacional, gran cantidad de memoria o ambos a la vez estos son los *clusters* utilizados. Se debe tener en cuenta que para la realización de

dichas tareas puede suponer la utilización de recursos del *cluster* durante largos periodos de tiempo.

- Alta disponibilidad (HA – High Availability): La base de estos *clusters* es disponibilidad y confiabilidad del sistema. Se garantiza la disponibilidad de los recursos ofrecidos junto con la detección de fallos y recuperación ante éstos en el software y la elusión de un solo punto de fallo en el hardware.
- Alta eficiencia (HT – High Throughput): Se pretende que la ejecución del mayor número de tareas se realice en el menor tiempo posible y el retardo de la comunicación entre los nodos del *cluster* no se considera un grave problema.

A continuación se enuncian los componentes que forma un *cluster* ilustrados en la Figura 5:

- Conexión de red: Necesaria para que los nodos del *cluster* pueda comunicarse entre ellos.
- Sistema operativo: Las características que debe tener cualquier sistema operativo presente en un *cluster* es ser *multiproceso* o *multiusuario*. Además, otra característica deseable, que no obligatoria, es que sea fácil de acceder y de utilizar.
- Nodos: Son las máquinas utilizadas para la formación del *cluster*.
- Middleware: Es el software que actúa de intermediario entre el sistema operativo y las aplicaciones. El objetivo es que el *cluster* se comporte como una sola entidad ante el usuario, ya que se tiene una única interfaz de acceso. Este software además tiene la función de realizar el balanceo de carga, la optimización del sistema o controlar la tolerancia a fallos entre otras y detecta la incorporación de un nodo nuevo potenciando la capacidad de escalado.
- Sistema de almacenamiento: Se puede hacer uso del almacenamiento interno de las máquinas de forma similar a la utilización de los discos duros de los PCs o se puede recurrir a sistemas de almacenamiento más complejos con mayor disponibilidad o eficiencia (NAS - Network Attaches Storage o SAN - Storage Area Network).

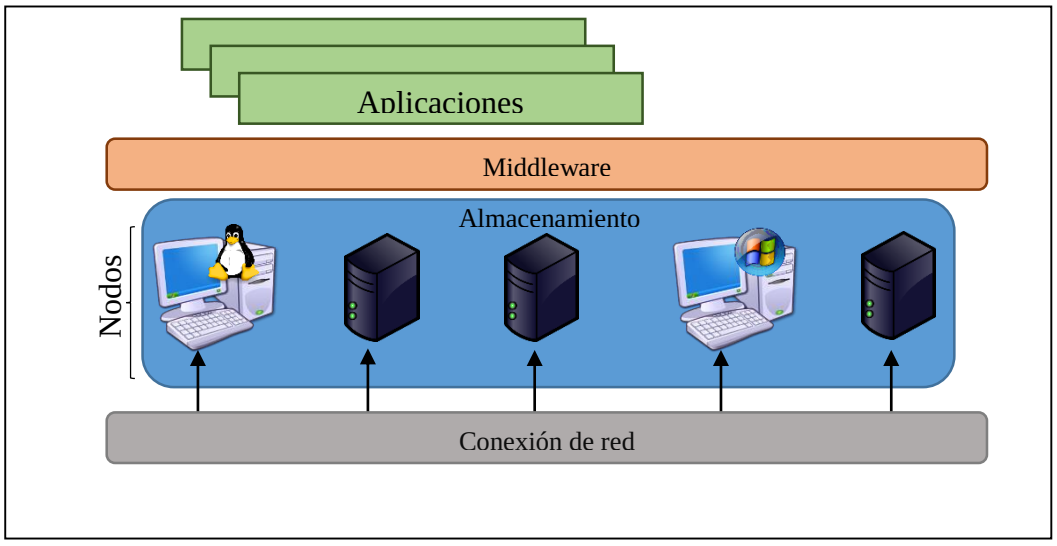


Figura 5. Componentes de un cluster.

2.2.3. Teorema de CAP

El teorema de CAP o de Brewer establece que en un sistema distribuido no puede garantizarse al mismo tiempo: consistencia, disponibilidad y tolerancia a fallos [30]. Estas tres propiedades pueden definirse según el teorema como [31]:

1. Consistencia: Se garantiza que en todos los nodos del sistema prevalezca la misma información. Para ello, la ejecución de un conjunto de operaciones se realiza al mismo tiempo quedando bloqueada la modificación de la información hasta la finalización de las mismas.

2. Disponibilidad: Esta propiedad se relaciona con el tiempo de respuesta del sistema. Cuando se excede el umbral establecido ya sea en función del tiempo de *timeout* de un socket o de la consideración de la espera que debe tolerar un usuario por ejemplo, el sistema se considera no disponible.

3. Tolerancia a fallos: A pesar de la pérdida de conexión de uno de los nodos, las peticiones de los usuarios siguen siendo procesadas por el resto de nodos del sistema.

Como se muestra en la Figura 6 sólo pueden darse dos de las tres propiedades del teorema de CAP por lo tanto el sistema puede asegurar [30]:

- CA: El sistema responderá a las peticiones en un tiempo de respuesta aceptable con la información modificada en el caso de que haya sufrido una actualización. Sin embargo, no puede permitirse la desconexión de cualquiera de los nodos que forman el sistema.
- CP: Se puede garantizar la consistencia del sistema de tal forma que todos los nodos poseen la misma información a pesar de que uno de los nodos pierda la conexión, pero el tiempo de respuesta puede exceder el umbral.
- AP: La pérdida de conexión de uno de los nodos no supondrá la caída de todo el sistema, además todas las peticiones se responderán sin exceder el tiempo de respuesta máximo establecido. En este caso, se pone en riesgo que la información no esté actualizada.

Figura 6. Teorema de CAP.

2.3. Infraestructura

2.3.1. MapReduce

Mapreduce [32] se define como una técnica de procesamiento y un modelo de programación para sistemas distribuidos basado en Java, diseñada por Google en 2004 y popularizada por Apache Hadoop (detallado más adelante) [33]. Este *framework*[1] realiza dos tareas, mapear y reducir. El mapeo se encarga de dividir un gran conjunto de datos en otros más pequeños, en los que los datos están formados por pares de clave-valor (tuplas). En el caso de la tarea de reducir, se recogen las divisiones resultantes de mapeo, las combina y las convierte en un conjunto de datos todavía de menor tamaño.

La descomposición del mapeo y la reducción en una aplicación es bastante complicada. Sin embargo, una vez programadas la aplicación puede ejecutarse en el número de nodos que se desee modificando la configuración de la ejecución. Por tanto una de las ventajas principales de esta técnica es que permite el procesamiento en múltiples nodos siendo compatible con la escalabilidad del sistema [34].

Para el entendimiento de las fases, se va a explicar con un contador de palabras en un texto (véase Figura 7):

1. Entrada y separación: El texto plano se convierte en una lista clave-valor, donde la clave puede ser el separador de una frase como un punto y el valor contiene las palabras de esa línea. Las listas generadas se distribuirán entre los diferentes nodos del *cluster*.
2. Mapeo: En esta fase se forman claves y valores intermedios, cada *cluster* tendrá su nueva lista generada en este proceso. Por cada palabra que se encuentre se crea una clave con el la palabra y el valor a un 1 [33].

$$\text{map } (K1,V1) \longrightarrow \text{lista } (K2,V2)$$

3. Combinación: Cada nodo que va a realizar la reducción tiene que contener las tuplas que comparten la misma clave, es decir, las tuplas con la misma palabra.
4. Reducción: Esta fase se encargará de hacer el conteo final del número de apariciones totales de cada palabra [33].

$$\text{reduce } (K2,\text{lista}(V2)) \longrightarrow (K3, V3)$$

5. Salida: Se juntan los resultados obtenidos en la reducción.

[1] Framework: Es una estructura conceptual y tecnológica de soporte definido mediante una serie de módulos de software que sirven de base para la organización y el desarrollo software.

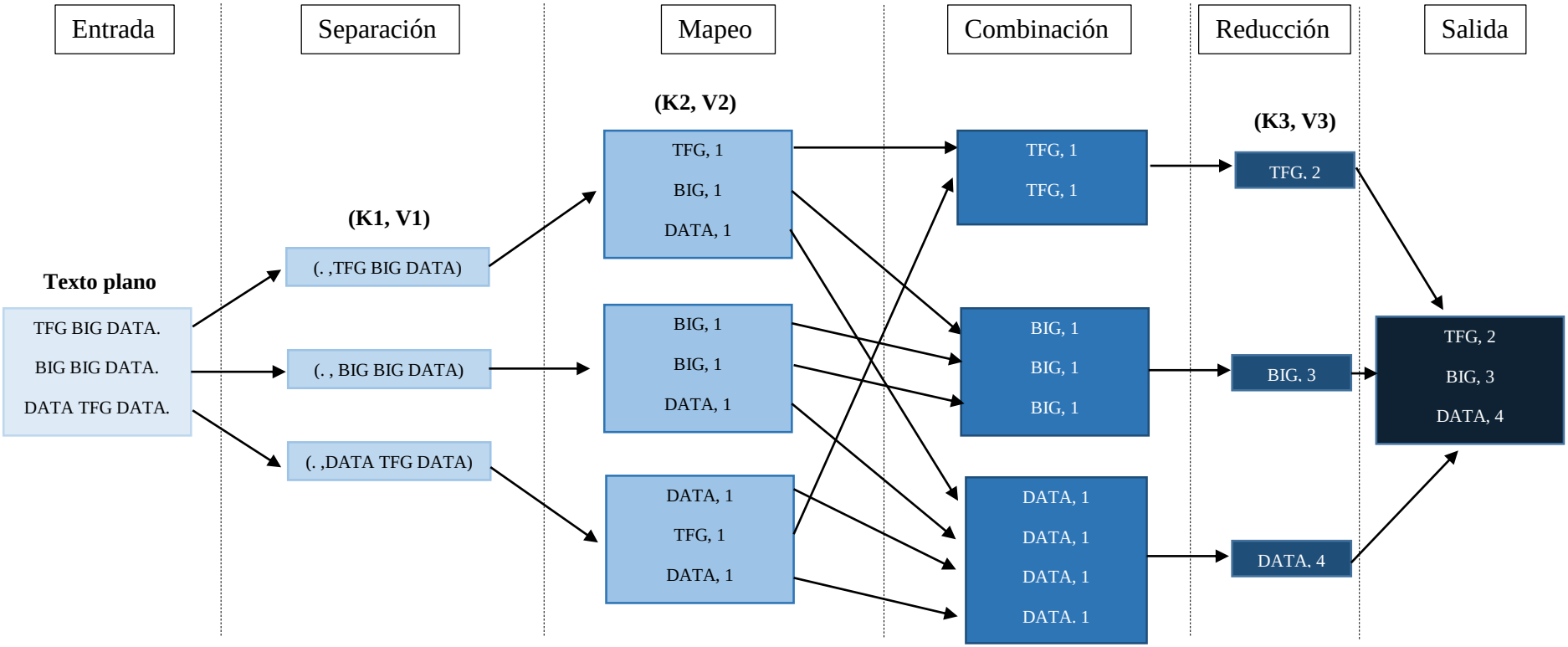


Figura 7. Ejemplo de Mapreduce.

2.3.2. Apache Hadoop

2.3.2.1. Introducción

Apache Hadoop [35] es un *framework* de código abierto desarrollado en lenguaje Java que permite el procesado de forma distribuida de grandes conjuntos de datos. Este procesado se realiza en los nodos que forman el *cluster* añadiendo potencia de cálculo y proporcionando un mayor almacenamiento [36].

El procesamiento de datos a gran escala requiere de un almacén capaz de contener grandes cantidades de datos y una forma de procesarlo eficaz. Estas dos necesidades las suplen los dos componentes principales del núcleo de *Hadoop* [36]:

- HDFS (Hadoop Distributed File System): Es un Sistema de ficheros distribuido basado en Google File System (GFS) y preparado para que la ejecución se realice en grandes *clusters*. Tiene una arquitectura maestro-esclavo, habiendo un único maestro (NameNode) y tantos esclavos (DataNodes) como se precise.
- Hadoop Mapreduce: Es un *framework* utilizado para procesar volúmenes elevados de datos de forma paralela en *clusters*. Como se ha detallado en la sección anterior (véase 2.3.1), tiene una primera tarea de mapeo en la que se convierte un conjunto de datos en una lista de clave-valor y una segunda de reducción que combina el conjunto de tuplas resultantes del mapeo y la combinación en conjuntos de datos clave-valor más pequeños.

Además de los componentes mencionados anteriormente, *Hadoop* está formado por dos componentes más que son Hadoop common que son las librerías de Java y las utilidades que necesitan otros módulos de *Hadoop* y Hadoop Yarn encargado de la planificación de tareas y gestión de todos los recursos del *cluster*.

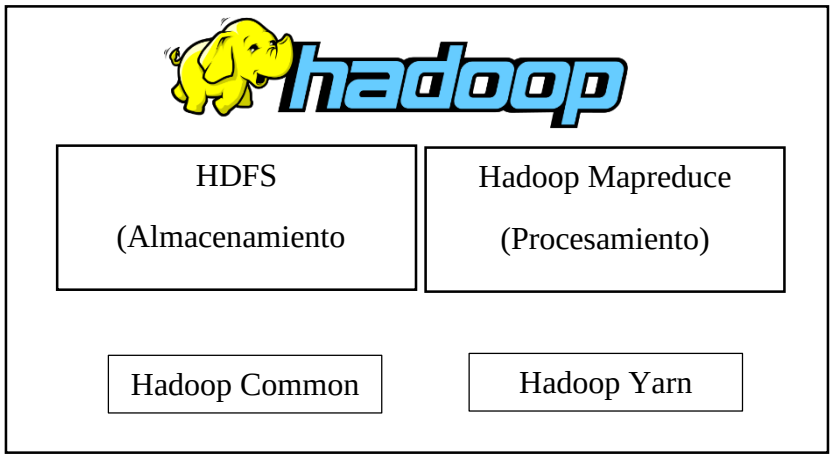


Figura 8. Estructura de Hadoop.

2.3.2.2. Demonios

Para el funcionamiento de *Hadoop* existen cinco o siete demonios[2], en función de la versión del *Mapreduce,* que son los siguientes [37]:

- Demonios de HDFS:
    1. NameNode: Es el encargado de los metadatos del sistema de ficheros HDFS.
    2. Secondary NameNode: Su función es la del mantenimiento del NameNode, pero no realiza funciones de backup.
    3. DataNode: Almacena los datos que han sido divididos en bloques en HDFS.
- Demonios de Mapreduce v1:
    1. JobTracker: Se encarga de la administración de tareas de *Mapreduce* y las reparte en tareas individuales entre los nodos del *cluster*.
    2. TaskTracker: Su objetivo es la supervisión de las tareas de mapeo y reducción.
- Demonios de Mapreduce v2 (Yarn):
    1. ResourceManager: Gestiona los recursos e inicia las tareas. Existe uno en cada cluster, no por nodo.
    2. ApplicationMaster: Ejecuta las tareas que han sido iniciadas por el ResourceManager. Hay uno por cada tarea iniciada.
    3. NodeManager: Es el que realiza las funciones de mapeo o reducción. Existe uno por cada nodo esclavo.
    4. JobHistory: Almacena las estadísticas y los metadatos del *cluster*.

2.3.2.3. Ventajas

La utilización de *Hadoop* permite al usuario probar de forma rápida un sistema distribuido debido a tu eficiencia a la hora de trabajar con varias máquinas en paralelo. Además garantiza la tolerancia a fallos y la alta disponibilidad siendo independiente del hardware empleado, por lo que permite la introducción o eliminación de nodos nuevos sin interrumpir el buen funcionamiento del *cluster*. Por último, una de sus ventajas más importantes, aparte de ser de código abierto, se ve reflejada en la compatibilidad con todas las plataformas ya que está desarrollado en Java.

[2] Demonio: En informática, es un proceso especial no interactivo, es decir, se ejecuta en segundo plano y no requiere de la intervención por parte del usuario.

2.3.2.4. HDFS

HDFS se encarga de almacenar los datos del *cluster*, y a diferencia de otros sistemas de ficheros distribuidos es muy tolerante a fallos (véase 2.2.3) y está diseñado con *hardware* de bajo coste [38].

El almacenamiento de los datos se hace entre los nodos del *cluster*. Los datos son divididos en bloques de 64MB y se almacenan en varias máquinas, es decir, los bloques se replican en diferentes nodos para garantizar la tolerancia a fallos ya que si uno pierde la comunicación los datos se encuentran en otros nodos [37].

Los efectos de la replicación de datos son un aumento en la fiabilidad del sistema ya que hay varias copias de los datos, mayor disponibilidad al encontrarse dichas réplicas en varios nodos del *cluster*, y aumento el rendimiento porque al haber más datos de puede realizar con mayor eficacia el procesamiento de éstos [37].

Por tanto las características de HDFS [38] se centran en la capacidad de almacenamiento y el procesamiento de datos que proporciona. Además, ofrece una interfaz de comandos amigable para el usuario con la que puede interactuar con HDFS e incluso conocer el estado actual del *cluster* en cada momento. También permite el acceso a los datos de forma instantánea, continua y sin interrupción y proporciona permisos de ficheros y autentificación. Sin embargo, existen inconvenientes en su utilización como la latencia que ocasiona la comunicación entre los nodos, las modificaciones siempre se realizan al final o que se trata de un modelo *single-writer multiples-readers*, es decir, no se permite la existencia de múltiples escritores [39].

En cuanto a la arquitectura (maestro- esclavo), consta de los siguientes elementos (véase Figura 9) [38]:

- NameNode: Es el maestro encargado de la administración del espacio de nombres de sistema de ficheros, controla el acceso de los clientes a los ficheros y ejecuta las operaciones relacionadas con la apertura y cierres de fichero y archivos o la modificación de nombres.
- DataNode: Realiza las operaciones de escritura y lectura a petición del cliente y crea, borra o replica los bloques siguiendo las instrucciones del NameNode.
- Bloques: Son las divisiones de los datos, es decir, la mínima cantidad de datos contenida en HDFS.

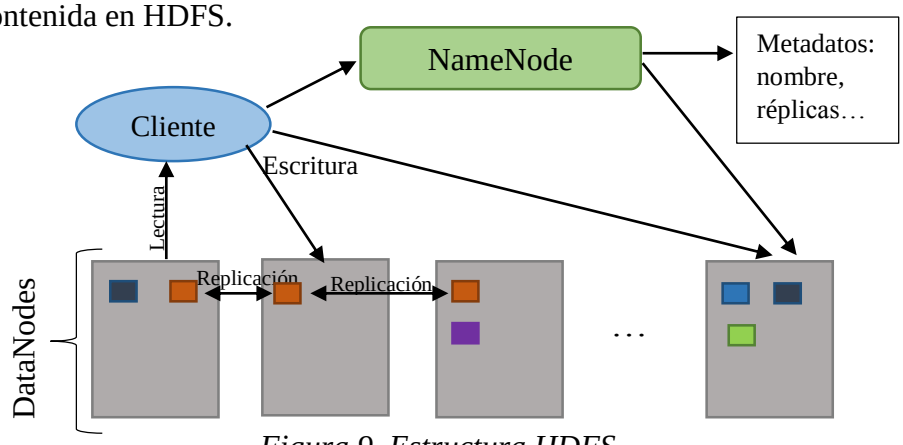


Figura 9. Estructura HDFS.

A continuación se van a detallar los pasos de lectura y escritura de un fichero en HDFS [37].

Lectura de un fichero

Los pasos que se siguen en HDFS para la lectura de un fichero son los siguientes:

1. En primer lugar el cliente es quien conecta con el NameNode para saber que DataNodes contienen los bloques que le interesa.
2. El NameNode devuelve una lista con dichos DataNodes.
3. El cliente se comunica con los DataNodes de la lista para proceder a la lectura del fichero.

Escritura de un fichero

Para la escritura de ficheros en HDFS se siguen los siguientes pasos:

1. Como en el caso de la lectura, el cliente conecta con el NameNode.
2. El cliente recibe el nombre del bloque y la lista de los DataNodes del sistema una vez el NameNode ha consultado en los metadatos.
3. Empieza el envío de datos desde el cliente al primer DataNode, y así hasta el último.
4. Se finaliza el envío por parte del cliente.
5. El cliente le indica al NameNode en que DataNodes se ha realizado la escritura del fichero.

Por último destacar lo que ya se lleva comentando con anterioridad, los objetivos de HDFS son detectar fallos y recuperarse ante los mismos, contar con un *cluster* formado por gran cantidad de nodos y ejecutar así de forma eficaz las tareas solicitadas [38].

2.3.3. Apache Hive

Los orígenes de *Hive* [40] provienen de Facebook ya que fue su creador. Este *framework* permite trabajar con el sistema distribuido de ficheros de *Hadoop* y facilita el manejo de datos [41].

El dialecto de *Hive* es similar al SQL denominado HQL, por lo que su funcionamiento se basa en consultas sobre los grandes volúmenes de datos almacenados en HDFS. Las consultas se traducen en trabajos *MapReduce* (véase 2.3.1) por lo que *Hive* nos facilita y abstrae de la compleja programación de las tareas de mapeo y reducción sobre los datos [41].

Hive no se considera una base de datos, pero tiene aspectos en común como la necesidad de crearnos un esquema de tablas con sus columnas y sus tipos para trabajar con los datos o cargar y realizar consultas sobre los datos para obtener los registros que deseemos. Sin embargo, existen dos inconvenientes a considerar, la traducción de la consulta al lenguaje Java para crear la tarea de *Mapreduce* aumenta el tiempo de respuesta y tampoco se pueden realizar todas las consultas que con SQL, además de no soportar índices ni transacciones.

2.4. AWS

2.4.1. Informática en la nube

La informática en la nube proporciona recursos informáticos y aplicaciones bajo demanda comunicados a través de internet y cuyo precio total de uso se realiza en función del consumo [42].

La *nube* es capaz de ofrecer a bajo coste acceso rápido y flexible a una serie de recursos sin necesidad de invertir en la adquisición de equipos ni tener la preocupación de la gestión de éstos. Por lo que, las facilidades que brinda la informática en la *nube* se basa en el acceso inmediato a los recursos y el pago sólo por el uso de los mismos [42].

Amazon Web Services (AWS), se encarga del mantenimiento de equipos mientras que el usuario únicamente aprovisiona los recursos a través de una aplicación web. El conjunto de servicios que ofrece se basan en servicios de informática, almacenamiento, bases de datos, análisis, aplicaciones e implementaciones que facilitan el avance de desarrollos, reduce los costes y la escalabilidad de las aplicaciones [42].

2.4.2. Productos y servicios

AWS ofrece numerosos servicios y productos, algunos de ellos son los que se muestran en las siguientes secciones (véase Figura 10).

2.4.2.1. S3

Amazon Simple Storage Service (Amazon S3), ofrece la capacidad de almacenamiento de forma segura, duradera y fuertemente escalable. Se caracteriza por su fácil utilización tanto a la hora de almacenar, como a la de recuperar datos desde cualquier ubicación en la web. Además proporciona distintos tipos de configuración referente al tiempo de almacenaje en función del acceso a los datos [43].

2.4.2.2. EC2

Amazon Elastic Compute Cloud (Amazon EC2), se trata de un servicio web que ofrece capacidad informática con tamaño modificable en función de las necesidades del usuario. Se establece control absoluto de sus recursos y permite ser ejecutado en entornos de Amazon y reduce el tiempo de obtención y arrancado de instancias de servidor [44].

2.4.2.3. EMR

Amazon Elastic MapReduce (Amazon EMR), está relacionado con el procesamiento a mayor velocidad de grandes cantidades de datos gracias a un *framework* de *Hadoop* gestionado. Esto ofrece la capacidad de distribuir y procesar los datos entre instancias EC2 que pueden ser escalables de forma rápida y sencilla [45].

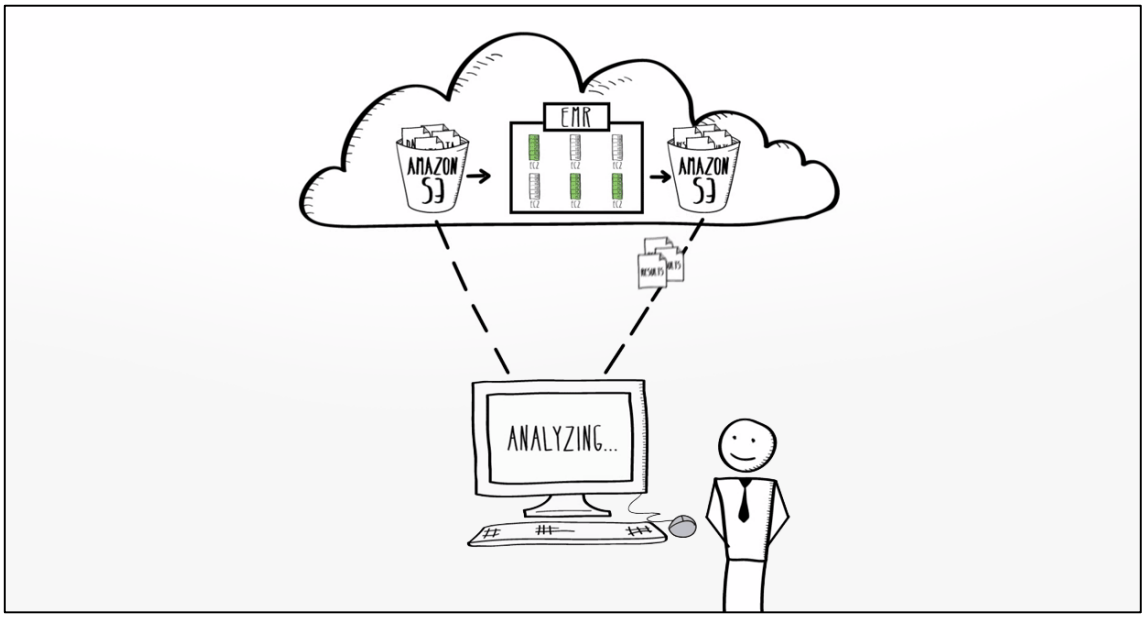


Figura 10. Interacción de S3, EC2 y EMR. (Figura tomada de [45])

Bloque III

Trabajo realizado

Capítulo 3

3. Diseño del sistema

3.1. Introducción

Una vez hecho el análisis de las tecnologías que permitirán llevar a cabo la implementación de nuestro sistema *big data*, vamos a centrarnos en el diseño del mismo. En primer lugar se presentará una visión general del sistema detallando los tres núcleos de funcionamiento. Después, se describirán los componentes que integran el sistema para dos posibles escenarios, de los cuales se ha procedido a implementar sólo uno de ellos siendo el escenario elegido para el desarrollo del proyecto. Por último, se ilustrarán las interacciones entre los componentes para entender el funcionamiento interno del sistema.

3.2. Núcleos de funcionamiento

El sistema *big data* sólo cuenta con un rol de usuario, el de administrador del sistema. Aun así, se podría restringir el acceso a cualquiera de las funcionalidades dependiendo del tipo de usuario. En este proyecto, no se ha considerado relevante esa distinción de roles ya que depende de quién explote el sistema y cómo lo explote, es decir, no hay roles definidos y fijos para actuar sobre cada una de las partes.

En nuestro diseño, teniendo en cuenta lo mencionado anteriormente acerca de los roles, el administrador del sistema tiene la capacidad de realizar el proceso de carga, el procesamiento y la visualización de los datos. La descomposición de funcionalidades que caracterizan el sistema dan lugar a los tres núcleos de funcionamiento del sistema (véase Figura 11):

1. Carga de datos: El usuario es capaz de cargar al sistema los archivos de datos con las trazas a procesar, en este proyecto las trazas corresponden a viajes de taxis en Nueva York. Por lo general, estos datos pueden proceder de distintas fuentes, aunque no es nuestro caso, por lo que no se puede garantizar la homogeneidad inicial en los datos tanto en formato como en estructura.

2. Procesado de datos: Los datos deben sufrir un proceso de trasformación y limpieza. En este núcleo se trata de dar formato a los datos y desechar aquellos que no son válidos mediante reglas de negocio y manipulaciones requeridas por el sistema de destino. Los registros provenientes de los taxis no tienen transformaciones directas sobre los campos de la traza, pero si se crean campos nuevos a partir de los ya existentes y se eliminan los registros considerados no válidos.
3. Visualización de datos: La representación de los datos posterior a su tratado puede seguir dos vía de interpretación. Se puede tener una representación de resultados obtenidos en su procesamiento, es decir, se pude buscar el resultado antes de la representación o se puede proceder a la explotación del dato para obtener conocimiento no trivial lo que se denomina *minería de datos* (véase 2.1.5). Uno de los objetivos del proyecto es la visualización de los datos tras su procesado, en concreto, se desean ver las diez rutas más frecuentes para cada hora de cada día, cada medio día de cada día y para cada día. Por esta razón, entendemos que nos encontramos en la primera vía de representación (resultados pre-visualizados).

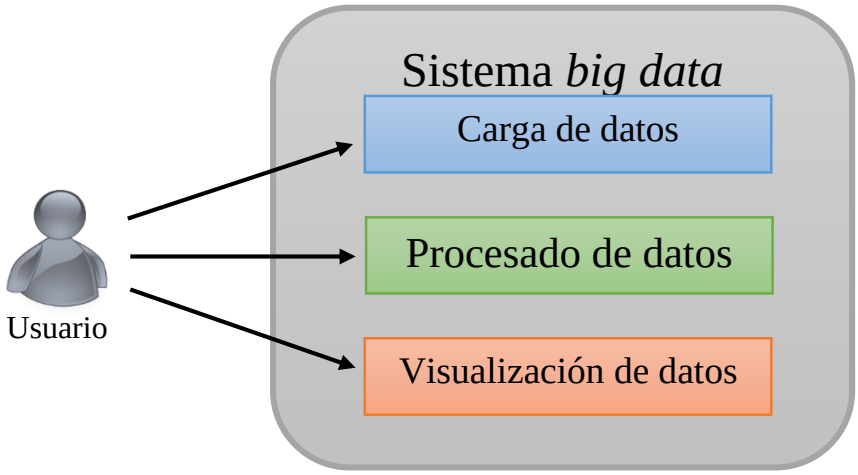


Figura 11. Esquema general del sistema big data con los núcleos de funcionamiento

3.3. Módulos del sistema / Componentes del sistema

El sistema precisará de una serie de componentes que deben relacionarse entre sí para dar continuidad a su flujo de actividad. Se muestran dos escenarios con los componentes involucrados dentro de cada núcleo de funcionamiento. Además, para entender el conjunto de módulos que forman el sistema se debe tener en cuenta la utilidad o finalidad del mismo.

Los componentes comunes a ambos sistemas son: *extracción de datos, introducción de datos, almacén de datos, cluster y pintado de datos* (véase Figura 12 y Figura 14). Sin embargo, sólo dos de ellos se comportan de la misma forma en ambos sistemas:

- Extracción de datos: Es el módulo encargado de la obtención de los datos desde los distintos sistemas de origen. La consecuencia de que los datos provengan de diferentes fuentes el formato no está estandarizado, lo que supone la trasformación del dato. La procedencia de los datos de este proyecto es el concurso DEBS 2015 Grand Challenge [2], y es un único fichero con las trazas de los viajes de los taxis de Nueva York.
- Introducción de datos: Recoge la petición del usuario de qué datos deben cargarse en el *almacén de datos*. Sólo cargaremos un archivo con todos los datos correspondientes a los viajes.

Para cada uno de los escenarios se va especificar el funcionamiento de los componentes así como la interactuación con los demás, ya que la interacción de mismos componentes con otros puede ser diferente dependiendo de la finalidad del sistema.

En el primer escenario (véase Figura 12), se quiere garantizar la persistencia del dato después de haber sido cargado y reformateado manteniéndolo en una base de datos. La repercusión en el sistema es la necesidad de incorporar el módulo de *persistencia del dato*, gracias al cual el usuario tiene la capacidad de explotar el dato extrayendo así información que no puede obtenerse a priori. Otra de las implicaciones es que la herramienta de visualización puede conectarse directamente a dicha base de datos. Los componentes para este escenario, además de *extracción de datos* e *introducción de datos* son:

- Almacén de datos: Guarda los datos que se quieren procesar y los datos resultados del procesado. El *cluster* se comunica con él para la carga inicial de los datos originales y proceder a la trasformación y eliminación de registros. Aunque también interactúa con el *cluster* para que éste deposite los datos una vez hayan sufrido el proceso de reformateo y limpieza. Además, el módulo de *persistencia de datos* asimismo conecta con el *almacén de datos* para mantener los datos una vez se han procesado.
- Cluster: Este componente lleva a cabo las tareas de procesamiento, es decir, la transformación y limpieza. Gracias a sus características (véase 2.2.2), permite mediante su configuración una mayor agilidad en el tratamiento de los datos. La interacción con el *almacén de datos* se ha explicado en el punto anterior. De lo que no se ha hablado todavía es de la relación con la *persistencia del dato.* Además de tener el resultado de procesado en el *almacén de datos,* otra opción sería mantener los datos directamente desde el módulo del *cluster* (véase Figura 13 b) en la *persistencia del dato*. También interactúa con el usuario, explicado en el último punto.
- Persistencia del dato: Base de datos que almacena los datos una vez han sido tratados bien a través del *almacén de datos* o mediante una conexión con el *cluster*. (véase Figura 13 a y Figura 13 b)
- Pintado de datos: En nuestro caso, con lo que se ha comentado anteriormente, tiene como función la representación de resultados que son las rutas más frecuentes en los distintos intervalos de tiempo. Sin embargo, se podrían pintar los datos sin buscar un resultado concreto e interpretarlos con ayuda de esta visualización, es decir, podríamos no saber que hay rutas frecuentes y una vez pintados todos los puntos veríamos a posteriori que efectivamente hay zonas de

mayor actividad. Se conecta con la *persistencia de datos* y poder así pintar los datos directamente sin intervención del usuario como se verá en el siguiente escenario (véase Figura 14).

- Usuario: Es el primer bloque, aunque no se engloba en el sistema *big data* es el bloque de arranque del sistema. Se tiene en cuenta que posee el rol de administrador del sistema, pero si se quisiera tener diferenciación de roles no se incorporaría ningún modulo adicional. Para incorporar distintos tipos de usuarios únicamente se validaría el rol de acceso por cada interacción directa que el componente *usuario* puede tener con el resto. Estas interacciones son con la *extracción de datos* para obtener los datos originales, con la *introducción de datos* para incorporar los datos origen, con el *cluster* para iniciar el procesado de los datos, con la *persistencia del dato* para hacer consultas con los datos y por último con el *pintado de datos* para ver la representación de los resultados de forma gráfica.

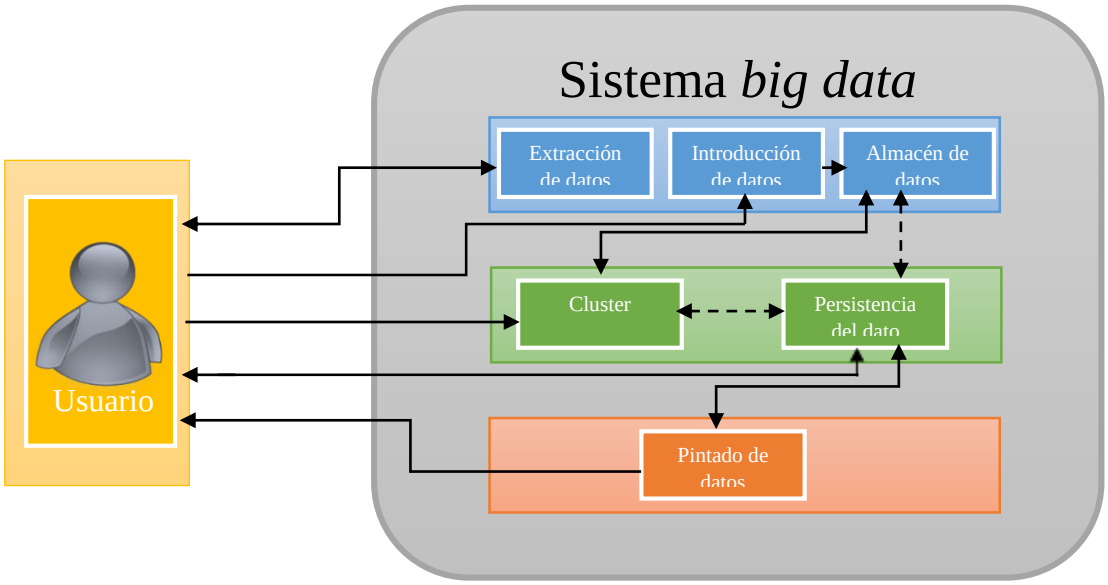
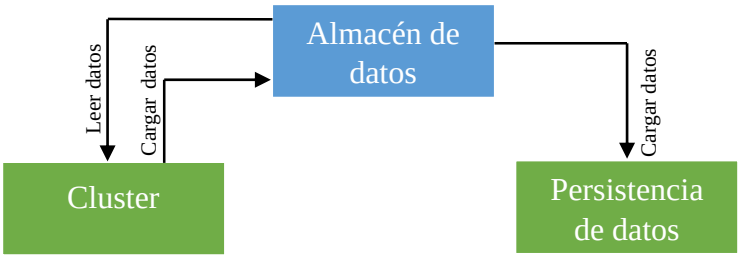
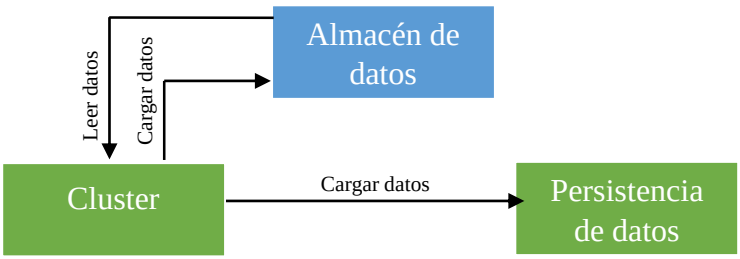


Figura 12. Escenario 1, interacciones y componentes del sistema big data.



a) Interacción de Almacén de datos con persistencia de datos



b) Interacción de Cluster con persistencia de datos

Figura 13. Interacción entre el cluster, el almacén de datos y la persistencia de datos.

A diferencia del escenario anterior, en este (véase Figura 14) se busca el resultado antes de la representación, no se requiere de la persistencia del dato porque no tiene como finalidad mantener los datos una vez han sufrido la transformación y limpieza y se incorpora el módulo de *obtención de resultados* para su posterior visualización. Además, el usuario debe cargar de forma manual los datos para poder proceder al pintado de datos. Los componentes que forman el segundo escenario, junto con la *extracción de datos* e *introducción de datos* son:

- Almacén de datos: Como en el escenario 1, es el componente encargado de guardar tanto los datos que se van a procesar, relacionados con los viajes realizados por los taxis en Nueva York, como los datos una vez han sido reformateados y limpiados. Este componente interactúa con el *cluster* para cargar los datos originales, reformatearlos, eliminar aquellos que no son válidos y almacenar los datos después de ser procesados. Otra línea de conexión es con el módulo de *obtención de resultados* para sacar los resultados y representarlos después.
- Cluster: La función es la misma que en el escenario anterior. Lleva a cabo las tareas de transformación y limpieza. Conecta con el *almacén de datos* para exportar los datos procesados y con el *usuario*, que se detallará más adelante.
- Obtención de resultados: Este componente hace de mediador entre el *almacén de datos* y el *usuario*, de tal forma que el usuario puede tener los datos ya preparados para ser pintando.
- Pintado de datos: La función es idéntica que en el escenario anterior. La diferencia es la conexión con los datos a la hora de ser representados. En este escenario, es el usuario quién introduce los datos para su visualización.

- Usuario: Se mantiene que el usuario solo tiene el rol de administrador, y para distinguir varios tipos de usuario se validaría el rol antes de interactuar con cualquiera de los componentes del sistema. El *usuario*, conecta con la *extracción de datos* para la obtención de los datos originales. También interactúa con el *almacén de datos* para guardar los datos de la extracción y con el *cluster* para iniciar el proceso de transformación y eliminación. Una vez se han procesado los datos, el *usuario* los recibe a través del módulo de *obtención de resultados.* Y por último, introduce los resultados en *pintado de datos* y los visualiza.

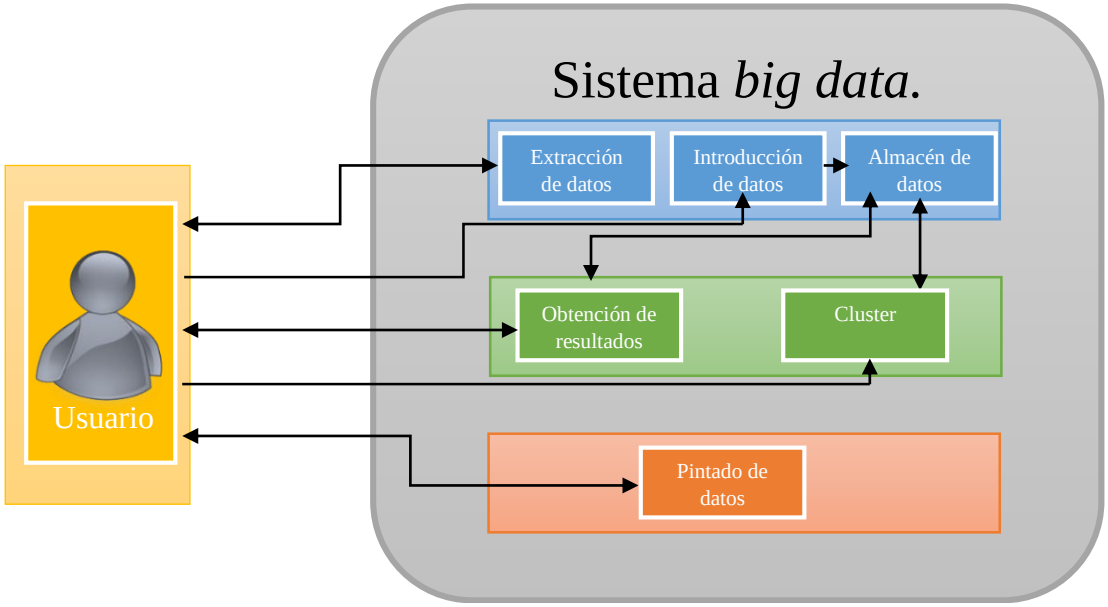


Figura 14. Escenario 2, interacciones y componentes del sistema big data.

3.4. Interacción de los componentes

Una vez introducidos los componentes fundamentales para la implantación del sistema *big data*, se describirá cómo colaboran entre ellos para conseguir el correcto funcionamiento del sistema. Se utilizará el diagrama de secuencia como herramienta gráfica para ilustrar las interacciones.

3.4.1. Carga de datos

En la Figura 15, se ilustra la interacción de componentes del primer núcleo de funcionamiento, que es igual para los dos escenarios diseñados en el apartado anterior. En primer lugar, se realiza la extracción de los datos originales del sistema de origen y después se introducen los datos en el *almacén de datos*.

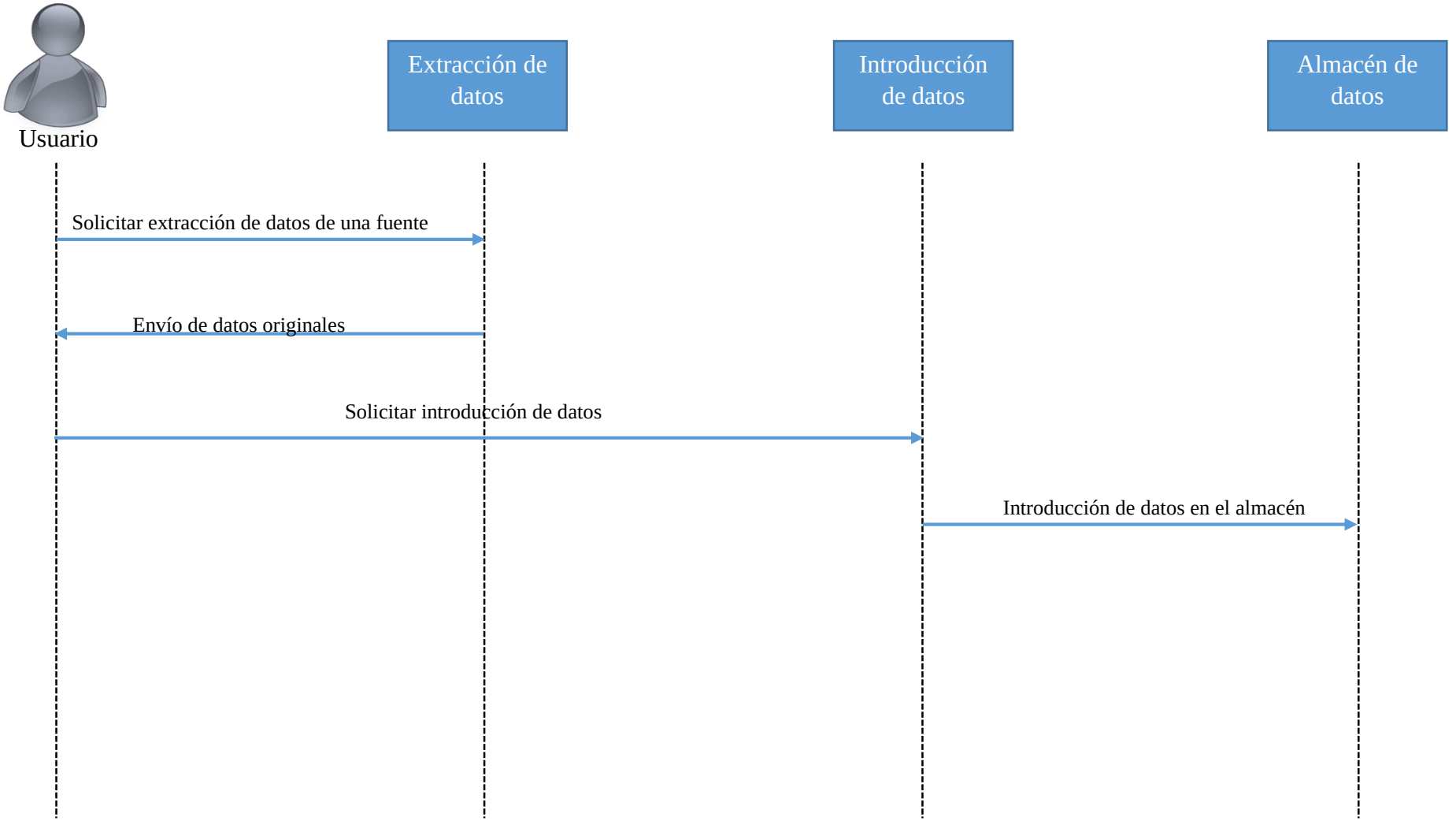


Figura 15. Diagrama de secuencia de la carga de datos.

3.4.2. Procesado de datos

En el procesado de datos los diagramas de secuencia para el escenarios 1 (véase Figura 16) y el escenario 2 (véase Figura 17) son distintos, pero tienen una parte en común. El usuario inicia el procesamiento de los datos que desencadena la acción de la recuperación de los datos originales y la recogida de los datos una vez que han sido transformados y limpiados.

Para el escenario 1, después de tener los datos procesados se tienen dos opciones para persistir el dato que depende únicamente de donde se lean. Una de ellas es cargar los datos del *almacén de datos*, y la otra obtenerlos del *cluster*. De esta forma, el usuario puede proceder a hacer consultas sobre los datos persistidos.

En el segundo escenario, el usuario simplemente obtiene los resultados después de solicitarlos. Estos resultados son los correspondientes a las diez rutas más frecuentes para cada hora de cada día, para los medios días de cada día y para cada día.

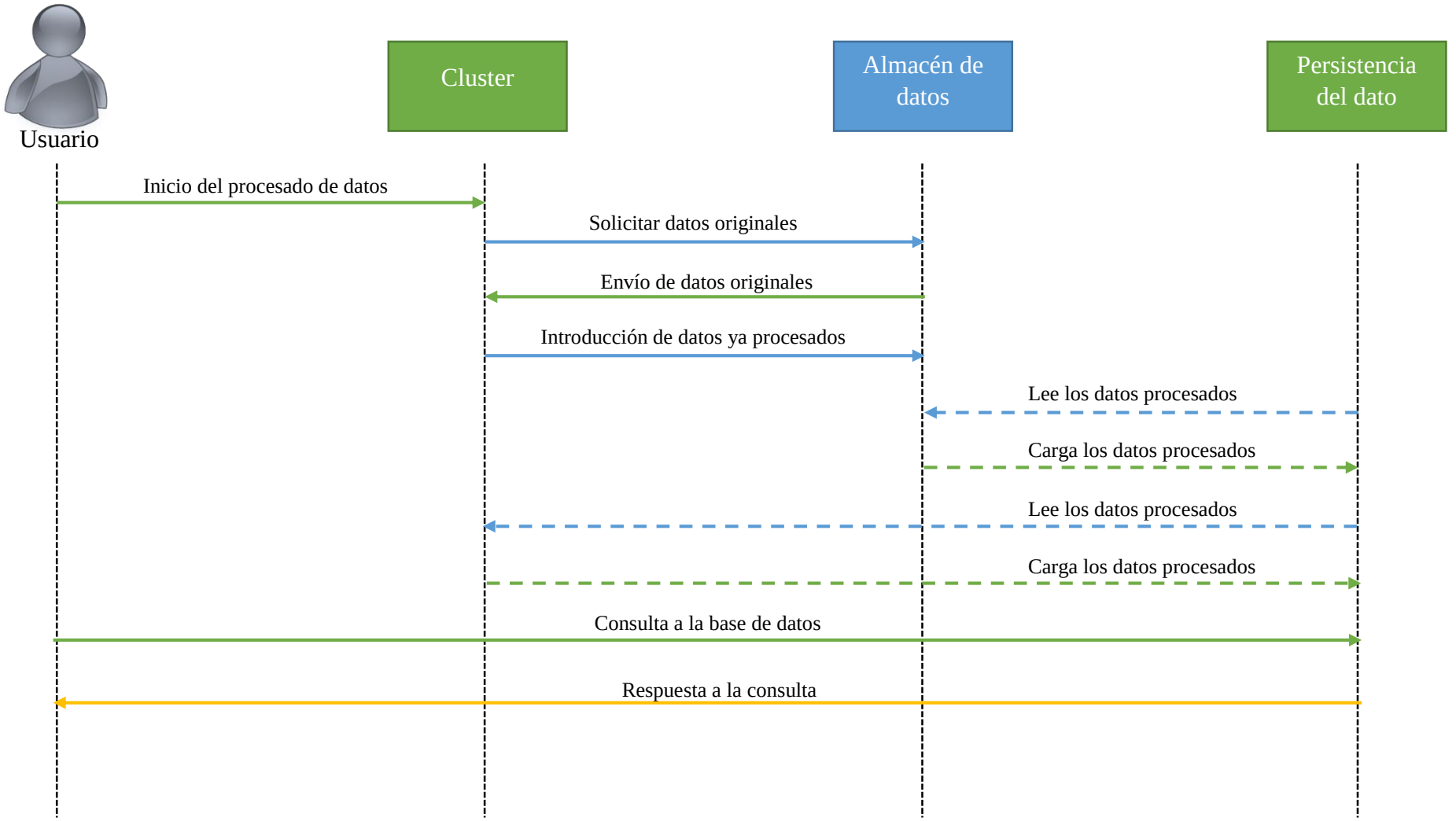


Figura 16. Diagrama de secuencia del procesamiento de datos en el escenario 1.

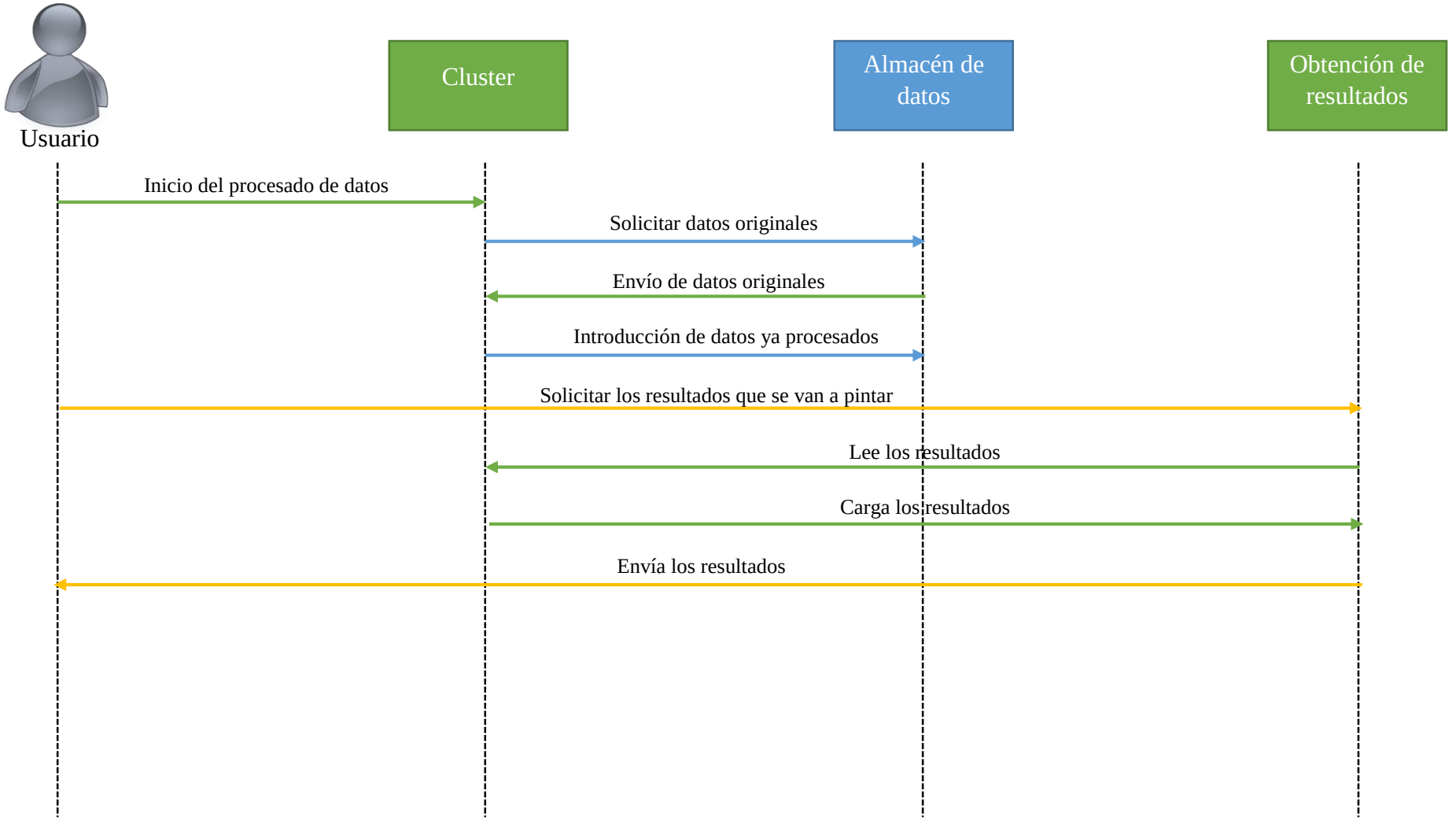


Figura 17. Diagrama de secuencia del procesamiento de datos en el escenario 2.

3.4.3. Visualización de datos

Dependiendo del escenario en el que nos encontremos la obtención de los datos que se quieren representar provienen de distinto sitio. En el primer escenario (véase Figura 19), el módulo de *pintado de datos* solicita los datos y se los muestra al *usuario*. Sin embargo, en el escenario 2 (véase Figura 18) el *usuario* introduce los datos en el de *pintado de datos* y ve la representación gráfica de éstos.

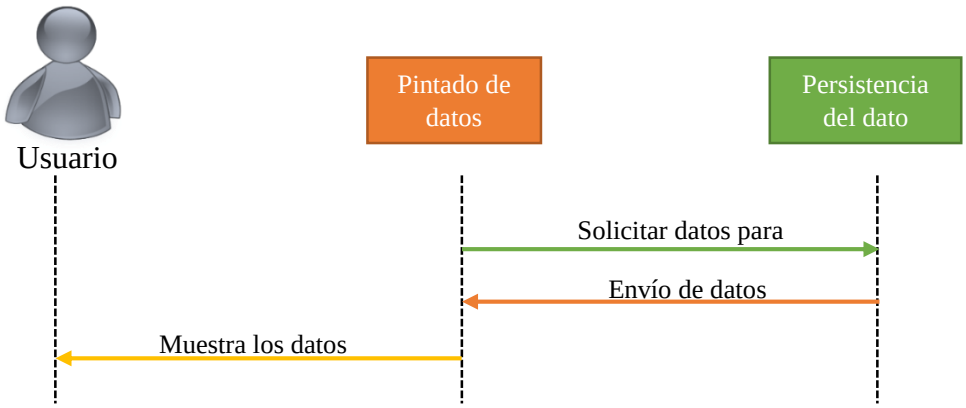


Figura 19. Diagrama de secuencia de la visualización de datos en el escenario 1.

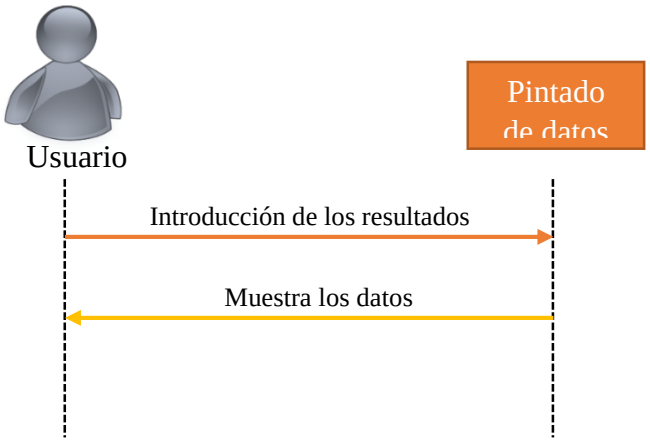


Figura 18. Diagrama de secuencia de la visualización de datos en el escenario 2.

Capítulo 4

4. Implementación

4.1. Introducción

En este capítulo se detallará los aspectos más importantes para la implementación, estableciendo una correspondencia entre los componentes del diseño y los elementos concretos de la tecnología empleada.

Como ya se ha comentado en el capítulo anterior, el escenario elegido para el desarrollo del proyecto es el escenario 2 (véase Figura 14). Al final es una simplificación del sistema para los requisitos establecidos. Las razones de la elección son las siguientes:

- No se ha incorporado la necesidad de mantener los datos en una base de datos, por lo que se prescinde del componente persistencia del dato que se encuentra en el escenario 1(véase Figura 12).
- Se ha establecido que los datos de entrada y de salida mantengan el mismo tipo de formato. Se espera la entrada de un fichero .csv y el mismo para la salida después del procesamiento.
- Se establecen los resultados que se quieren visualizar previamente a la representación. El usuario introducirá los resultados para ver las diez rutas más frecuentes en la herramienta de visualización, se prescinde de conectar dicha herramieta a una base de datos.

En primer lugar, se detallará cuál ha sido el entorno de desarrollo del proyecto y las instalaciones necesarias para su funcionamiento. Después se expondrá como se cargan los datos dentro del entorno, en nuestro caso la carga de las trazas correspondientes a los viajes de los taxis en Nueva York. Más adelante, se explicarán cuáles son los pasos a seguir para realizar el procesamiento de los datos y quedarnos así con los registros que consideramos válidos descartando aquellos que no lo son. Y por último, se describirá cómo introducir los resultados, que es nuestro caso son los datos correspondientes a las rutas más frecuentes, en la herramienta de visualización.

4.2. Entorno de implementación

El proyecto se va a implementar en una máquina virtual. La máquina virtual es un software que permite instalar en el ordenador, dónde se encuentra el sistema anfitrión, otros sistemas operativos o también llamados sistemas invitados. Esto nos posibilita la acción de trabajar con otros sistemas operativos como si fueran aplicaciones en nuestro ordenador. Además, cabe destacar que la máquina virtual utiliza los recursos hardware físico del ordenador y dichos recursos son administrador por el hypervisor, que es un sistema encargado de coordinar las instrucciones de CPU.

Para la creación de la máquina virtual que emplearemos para la implementación del sistema *big data* diseñado utilizaremos VirtualBox (véase Figura 20), que es el software de virtualización que nos permitirá trabajar con el sistema invitado.

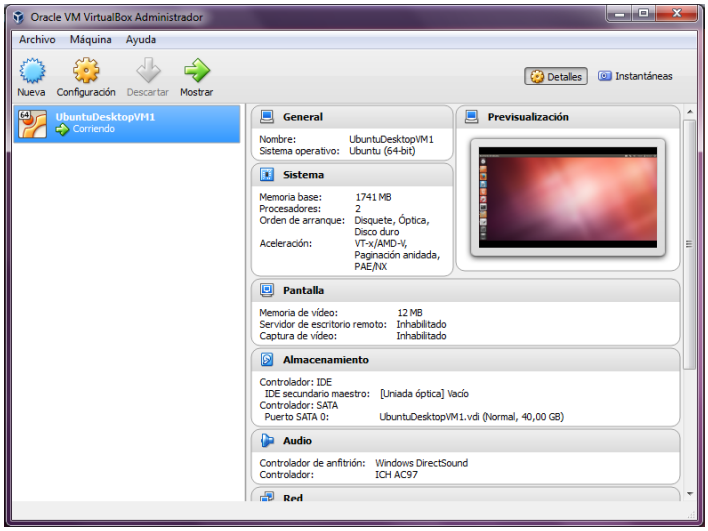


Figura 20. Máquina virtual en VirtualBox.

La elección de este entorno se debe a la capacidad de instalar el *almacén de datos* y el *framework* empleados para el desarrollo de este proyecto, ya que no existe la posibilidad de instalación directa en Windows (sistema operativo del ordenador utilizado para el proyecto).

4.3. Introducción de trazas

El primer objetivo es la carga de los datos en el sistema. Para ello, hemos conseguido las trazas de los viajes de los taxis en Nueva York a través de la página DEBS 2015 Grand Challenge [2]. Por lo que nuestros datos sólo provienen de un único sistema de origen y será un fichero que contiene todas las trazas.

A continuación de muestra en la Tabla 1, los campos de cada registro del fichero y un ejemplo del mismo en la Figura 21.

Campo	Descripción
id_taxi	Identificador del taxi
licencia_taxi	Identificador de la licencia del taxi
fecha_origen	Fecha de inicio del viaje
fecha_destino	Fecha del final del viaje
duracion_viaje	Duración del viaje (en segundos)
distancia_viaje	Distancia del viaje (en millas)
longitud_origen	Longitud del punto de inicio del viaje
latitud_origen	Latitud del punto de inicio del viaje
longitud_destino	Longitud del punto de fin del viaje
latitud_destino	Latitud del punto de fin del viaje
tipo_pago	Método de pago
cantidad_tarifa	Cantidad de la tarifa(en dólares)
recargo	Cargo adicional (en dólares)
mta_tasas	Tasas (en dólares)
propina_viaje	Propina
precio_tuneles	Peajes de puentes y túneles (en dólares)
precio_total	Cantidad total (en dólares)

Tabla 1. Campos de la traza y su descripción.

```
DFBFA82ECA8F7059B89C3E8B93DAA377,CF8604E72D83840FBA1978C2D2FC9CDB,2013-01-01 00:02:00,2013-01-
01 00:03:00,60,0.39,-73.981544,40.781475,-73.979439,40.784386,CRD,3.00,0.50,0.50,0.70,0.00,4.70
```

Figura 21. Ejemplo de la traza de viajes de taxis en NYC.

Una vez descargados los datos que vamos a tratar, los incorporaremos al *almacén de datos*. Este almacén es HDFS (véase 2.3.2.4) creado para trabajar con grandes volúmenes de datos. La interacción con el *de datos* se hace a través de la ventana de comandos de la máquina virtual, y a partir de ella podemos crear carpetas, cargar archivos y consultar el contenido de los directorios entre otras funcionalidades.

A continuación se va a mostrar la estructura de directorios dentro del *almacén de datos* (véase Figura 22) y esa estructura vista desde la consola de la máquina virtual (véase Figura 23). Se recuerda que para la utilización de los comandos sobre el *almacén de datos* y para el posterior tratado de los datos se necesita tener arrancados los demonios de HDFS (véase 2.3.2.2) (véase Figura 24).

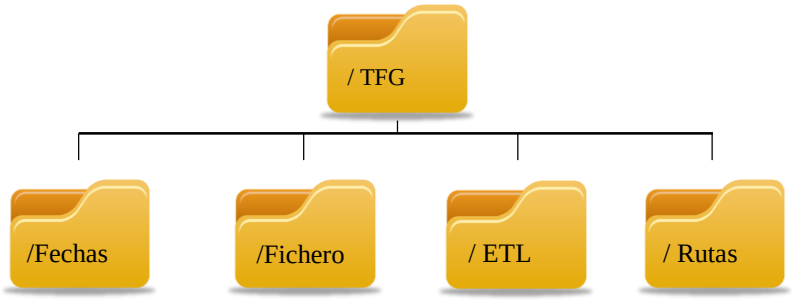


Figura 22. Estructura de directorios en HDFS.

Cada directorio tiene su finalidad dentro del proyecto, en concreto a la hora de procesar los datos. La función de cada uno de ellos es:

- /Fechas: En este directorio se cargarán los ficheros del control de tiempos para cada etapa del procesado.
- /Fichero: Será donde se almacene el fichero con los datos que se quieren procesar.
- /ETL: Recogerá los datos de salida, es decir, los datos una vez han sido procesados.
- /Rutas: Almacenará los archivos con los resultados de las rutas más frecuentes, que se utilizarán para pintarlos con la herramienta de visualización.

```
hduser@UbuntuVM:~/hadoop$ hdfs dfs -ls /TFG/
16/02/03 15:19:14 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 4 items
drwxr-xr-x   - hduser supergroup          0 2016-01-20 14:40 /TFG/ETL
drwxr-xr-x   - hduser supergroup          0 2016-01-18 18:29 /TFG/Fechas
drwxr-xr-x   - hduser supergroup          0 2016-01-19 17:06 /TFG/Fichero
drwxr-xr-x   - hduser supergroup          0 2016-01-20 14:45 /TFG/Rutas
```

Figura 23. Visualización de directorios desde la consola.

```
hduser@UbuntuVM:~/hadoop$ ./sbin/start-dfs.sh
16/02/03 15:14:45 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hduser/hadoop/logs/hadoop-hduser-
namenode-UbuntuVM.out
localhost: starting datanode, logging to /home/hduser/hadoop/logs/hadoop-hduser-
datanode-UbuntuVM.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/hduser/hadoop/logs/hadoop-
hduser-secondarynamenode-UbuntuVM.out
16/02/03 15:15:20 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
hduser@UbuntuVM:~/hadoop$ ./sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/hduser/hadoop/logs/yarn-hduser-resour
cemanager-UbuntuVM.out
localhost: starting nodemanager, logging to /home/hduser/hadoop/logs/yarn-hduser
-nodemanager-UbuntuVM.out
```

Figura 24. Comandos para arrancar los demonios HDFS.

La creación de carpetas se hace desde la ventana de comandos. Para los directorios de primer nivel, el comando empleado es *hdfs dfs –mkdir /Directorio1*. Y para los subdirectorios *hdfs dfs –mkdir /Directorio1/Directorio2*. En nuestro caso, el directorio del primer nivel es TFG por lo que el comando empleado es *hdfs dfs –mkdir/TFG*. En en la creación directorio de segundo nivel se emplearía, por ejemplo para Fechas, *hdfs dfs –mkdir /TFG/Fechas* e igual para el resto.

Una vez tenemos los directorios creados, sólo tenemos que cargar nuestro archivo en la carpeta correspondiente que es /TFG/Fichero. La carga se hace con el comando *hdfs dfs –copyFromLocal /LocalPath /HDFSPath* (véase Figura 25).

```
hduser@UbuntuVM:~/hadoop$ hdfs dfs -copyFromLocal /home/hduser/Documentos/TFG/
Ficheros/data.csv /TFG/Fichero
```

Figura 25. Comando para la carga de fichero en HDFS.

Después de seguir todos estos pasos, tenemos el archivo en el *almacén de datos* y podemos continuar con su procesamiento.

4.4. Tratado de los datos

Todo el tratado de datos se realiza con *Hive* (véase 2.3.3), que es un software de almacenamiento de datos que facilita las consultas y el análisis de grandes volúmenes de datos almacenados en HDFS. Esas consultas se pueden hacer sentencia a sentencia por línea de comandos, pero se ha realizado un script .hql que ejecutará todas las sentencias de forma secuencial. Para su ejecución desde la ventana de comandos de la máquina virtual se hace mediante el comando *hive –f script.hql* (véase Figura 26).

```
hduser@UbuntuVM:~/Documentos/TFG/Scripts/HIVE$ hive -f script.hql

Logging initialized using configuration in jar:file:/usr/lib/hive/apache-hive-0
.13.0-bin/lib/hive-common-0.13.0.jar!/hive-log4j.properties
OK
Time taken: 2.25 seconds
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1454508944121_0002, Tracking URL = http://UbuntuVM:8088/prox
y/application_1454508944121_0002/
Kill Command = /home/hduser/hadoop/bin/hadoop job  -kill job_1454508944121_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-02-03 15:35:15,238 Stage-1 map = 0%,  reduce = 0%
```

Figura 26. Ejecución script.hql

En la etapa del procesado de los datos, como se lleva exponiendo a lo largo del documento, consiste en la transformación de los datos de ser necesario y en la eliminación de los registros que no son válidos.

En nuestro caso, al no persistir el dato en una base de datos no se necesitan reformatear para que los datos sean aceptados por el sistema destino que sería la base de datos encargada de mantenerlos. El procesamiento va a consistir en eliminar aquellas trazas que hemos considerado erróneas para la obtención de los resultados. Además, la tabla final tendrá los campos que nos interesan y crearemos cuatro campos nuevos (véase Tabla 2).

Para la validación de las trazas los criterios que se han utilizado para descartar registros han sido los siguientes:

- El identificador del taxi no puede ser nulo. De serlo podría ser un taxi que está recogiendo datos erróneos.
- La licencia del taxi no puede ser nula. Por el mismo razonamiento anterior.
- La fecha de inicio del viaje y la fecha de fin del viaje no puede ser igual, ya que se trataría de un viaje nulo.
- La duración del viaje no puede ser menor que cero. El dato sería incorrecto.
- El tipo de pago no puede ser nulo. El taxi estaría recogiendo datos mal.
- El precio total debe ser mayor que cero, igual que la duración del viaje.

En la limpieza de registros también tenemos que tener en cuenta que las rutas más frecuentes van a definidas por cuadrados dentro de una cuadrícula de tamaño 300 x 300 (véase Anexo A: Cuadrícula para la visualización de rutas). El punto de inicio del viaje o el punto de fin del viaje no puede salirse fuera de ésta cuadrícula, por lo que se consideran erróneos aquellos registros cuyo identificador del cuadrado en el que se encuentren el origen o destino estén fuera de ese rango.

La eliminación de las trazas no válidas se hace en dos pasos. En primer lugar, se calcula el número del cuadrado (identificador del cuadrado) correspondiente del punto de origen y de destino y se eliminan los registros que no cumplen los seis primeros criterios establecidos (véase Figura 27). Después, se eliminaran aquellas trazas cuyo punto de partida o fin se salga de la cuadrícula (véase Figura 28).

```
CREATE TABLE aux_taxis_etl_1 AS
SELECT
        id_taxi,
        licencia_taxi,
        fecha_origen,
        fecha_destino,
        duracion_viaje,
        longitud_origen,
        latitud_origen,
        cast(ceil((((74.916578-(longitud_origen*-1)))/0.005986))as int) as id_cua_or_1,
        cast(ceil(((41.47718278-latitud_origen)/0.004491556))as int) as id_cua_or_2,
        longitud_destino,
        latitud_destino,
        cast(ceil((((74.916578-(longitud_destino*-1)))/0.005986))as int) as id_cua_des_1,
        cast(ceil(((41.47718278-latitud_destino)/0.004491556))as int) as id_cua_des_2,
        tipo_pago,
        precio_total,
        precio_total*0.898190147 as precio_total_euros
FROM taxis_original
WHERE   id_taxi <> "" AND
        licencia_taxi <> "" AND
        fecha_origen <> fecha_destino AND
        duracion_viaje > 0 AND
        tipo_pago <> "" AND
        precio_total > 0 AND
        longitud_origen <> longitud_destino AND
        latitud_origen <> latitud_destino;
```

Figura 27. Eliminación de registros según los primeros seis criterios.

```
CREATE TABLE aux_taxis_etl_2 AS
SELECT
        id_taxi,
        licencia_taxi,
        fecha_origen,
        fecha_destino,
        duracion_viaje,
        longitud_origen,
        latitud_origen,
        id_cua_or_1,
        id_cua_or_2,
        concat(concat(cast(id_cua_or_1 as varchar(20)),"."),cast(id_cua_or_2 as varchar(20))) as id_cuadricula_origen,
        longitud_destino,
        latitud_destino,
        concat(concat(cast(id_cua_des_1 as varchar(20)),"."),cast(id_cua_des_2 as varchar(20))) as id_cuadricula_destino,
        id_cua_des_1,
        id_cua_des_2,
        tipo_pago,
        precio_total,
        precio_total_euros
FROM aux_taxis_etl_1
WHERE   id_cua_or_1>0 and id_cua_or_1<300 and
        id_cua_or_2>0 and id_cua_or_2<300 and
        id_cua_des_1>0 and id_cua_des_1<300 and
        id_cua_des_2>0 and id_cua_des_2<300;
```

Figura 28. Eliminación de registros fuera del rango de la cuadrícula.

Tras la eliminación de registros y quedándonos con los campos que nos interesan además de añadir los nuevos, los registros tendrán la siguiente información:

Campos finales
id_taxi
licencia_taxi
fecha_origen
fecha_destino
duracion_viaje
~~distancia_viaje~~
longitud_origen
latitud_origen
longitud_destino
latitud_destino
tipo_pago
~~cantidad_tarifa~~
~~recargo~~
~~mta_tasas~~
~~propina_viaje~~
~~precio_tuneles~~
precio_total
id_cua_or_1
id_cua_or_2
id_cuadricula_origen
id_cuadricula_destino

Tabla 2. Campos de la tabla final.

Una vez filtradas las trazas, se procede al cálculo de las rutas por horas, medios días y días. Para ello se hace una agrupación de los registros dependiendo del periodo de tiempo elegido y un conteo de las veces que aparece cada cuadricula de origen y de destino quedándonos con los diez registros que más se repiten.

```
SELECT aux.*
FROM (SELECT ROW_NUMBER() OVER(PARTITION BY fecha_ruta,hora_ruta ORDER BY num_rutas DESC) as num,*
        FROM (SELECT date(fecha_origen) as fecha_ruta,hour(fecha_origen) as hora_ruta,cast(count(*) as int) as num_rutas,id_cuadricula_origen, id_cuadricula_destino
            FROM taxis_etl
            GROUP BY date(fecha_origen),hour(fecha_origen),id_cuadricula_origen, id_cuadricula_destino
            ORDER BY num_rutas DESC
            ) as aux_1
    ) as aux
WHERE num < 11;
```

Figura 29. Rutas más frecuentes por horas de cada día.

```
SELECT aux.*
FROM (SELECT ROW_NUMBER() OVER(PARTITION BY fecha_ruta ORDER BY num_rutas DESC) as num,*
        FROM (SELECT fecha_ruta, cast(count(*) as int) as num_rutas,id_cuadricula_origen,id_cuadricula_destino
                FROM(SELECT date(fecha_origen) as fecha_ruta,hour(fecha_origen) as hora_ruta,id_cuadricula_origen, id_cuadricula_destino
                        FROM taxis_etl
                    ) as aux_1
                WHERE hora_ruta >=0 AND hora_ruta <= 12
                GROUP BY fecha_ruta,id_cuadricula_origen,id_cuadricula_destino
                ORDER BY num_rutas DESC
            ) as a
    ) as aux
WHERE num < 11;
```

Figura 30. Rutas más frecuentes por medio día de cada día.

```
SELECT aux.*
FROM (SELECT ROW_NUMBER() OVER(PARTITION BY fecha_ruta ORDER BY num_rutas DESC) as num,*
        FROM(SELECT date(fecha_origen) as fecha_ruta,cast(count(*) as int) as num_rutas,id_cuadricula_origen, id_cuadricula_destino
                FROM taxis_etl
                GROUP BY date(fecha_origen),id_cuadricula_origen, id_cuadricula_destino
                ORDER BY num_rutas DESC
            ) as aux_1
    ) as aux
WHERE num < 11;
```

Figura 31. Rutas más frecuentes por día.

El último paso a realizar antes de la representación gráfica es obtener los resultados hasta ahora almacenados en HDFS. Los extraemos en un fichero con el comando *hdfs dfs –copyToLocal /HDFSPath /LocalPath*, y éste será el archivo que contenga la información de las rutas que vamos a proceder a visualizar.

4.5. Representación de rutas

Finalmente, después de almacenar los datos originales, de procesarlos y de tener las rutas más frecuentes se realiza la visualización de éstas últimas en un mapa. Para ello, se ha utilizado la herramienta QGIS [46], que es un sistema de información geográfica de código abierto.

Lo primero a tener en cuenta es que las rutas están definidas por un cuadrado de origen y un cuadro de destino dentro de la cuadrícula de 300 x 300. Esto quiere decir que no son punto de partida y de fin con una longitud y latitud exacta, sino que hay un rango de valores para la latitud y longitud de origen e igual para el destino (véase Anexo A: Cuadrícula para la visualización de rutas).

Para la visualización de esas rutas se ha cogido una traza de ejemplo por cada una de las diez rutas por hora, por medio día y por día, que tuviera como cuadrado de origen y de destino el mismo que el identificado como ruta más frecuente. Y se han insertado los puntos a través de un fichero de texto separado por tabulaciones en la herramienta. También es importante destacar que el periodo de tiempo para el que se han representado las rutas es el día 1 de Enero del 2013 en el que se recogen los tres periodos establecidos de las diez rutas más frecuentes.

Antes de introducir los puntos, se han instalado dos *plugins*[3] de la herramienta para la representación:

- OpenLayers plugin: permite cargar diferentes tipos de mapas en la interfaz gráfica y ofrece una visualización previa a la inserción de la capa. Algunos de los mapas que pueden elegirse pueden ser de OpenStreetMaps, Google Maps, Bing Maps, MapQuest, OSM/Stamen y Apple Maps.
- Online Routing Mapper: Permite calcular rutas a partir de un punto de origen y un punto de destino seleccionados en el mapa. Las rutas representadas pueden ser reales, es decir, este complemento utiliza el API de Google Direction por lo que se tiene consciencia de la dirección de las calles.

En mi primer lugar a través del complemento OpenLayers plugin hemos definido el mapa de OpenStreetMaps como primera capa. Después con ayuda de las herramientas de investigación hemos creado la cuadricula con el ancho y alto de los cuadrados partiendo de las longitudes y latitudes máximas y mínimas de la cuadrícula.

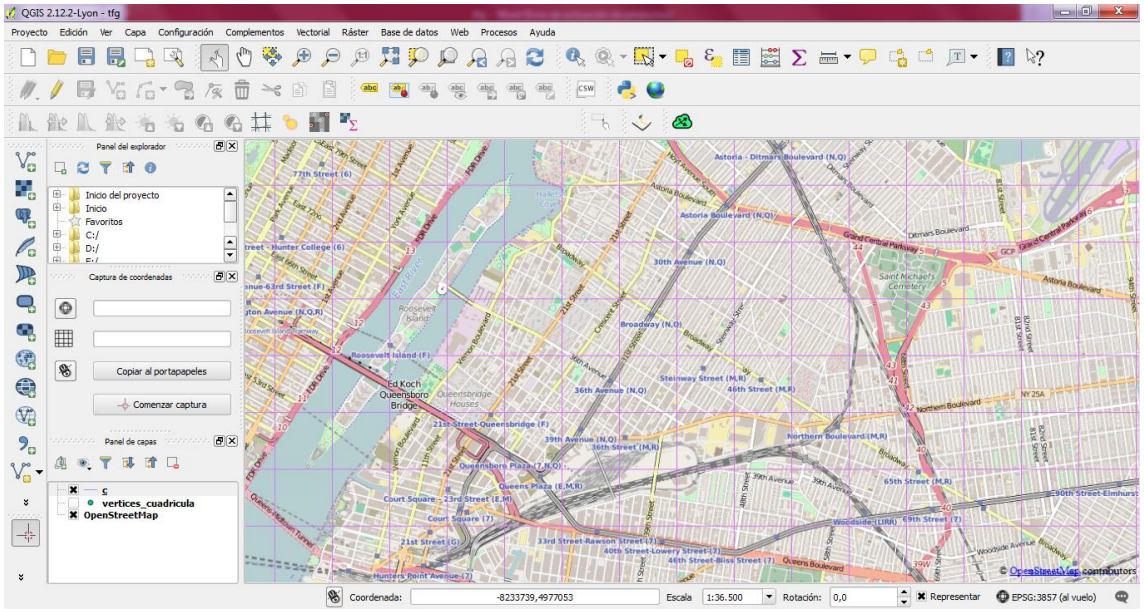


Figura 32. Visualización de la cuadrícula en el mapa.

[3] Plugin: es un complemento que puede añadir una funcionalidad o característica extra al programa informático en el que se aplique.

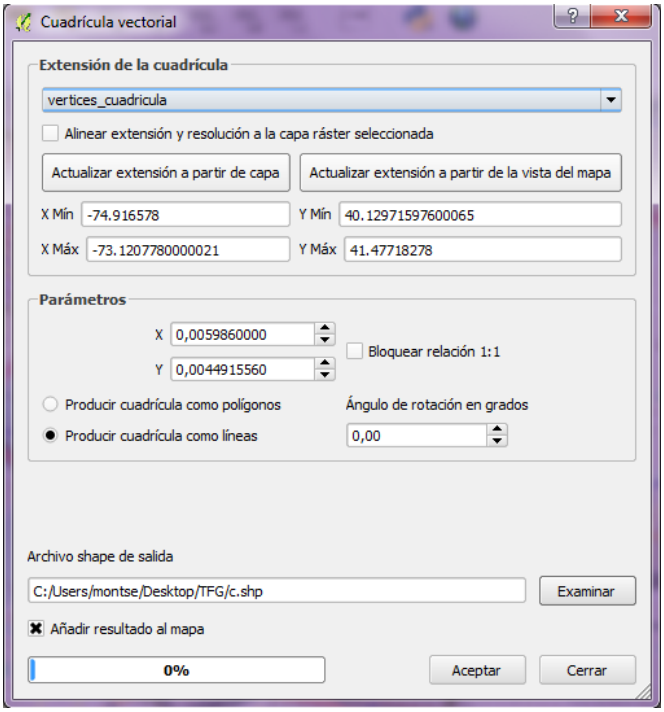


Figura 33. Generación de la cuadrícula.

Una vez representado el mapa con la cuadrícula, se insertan los puntos de ejemplo con la opción de QGIS "añadir capa de texto delimitado". Así tendremos todos los puntos tanto de origen como de destino de las diez rutas pero sin estar conectados.

Para la unión de los puntos, se utiliza el *plugin* de Online Routing Mapper. Este complemento nos permitirá seleccionar el punto de origen y de destino y dibujándonos la ruta entre ambos. Se recuerda que la información que nos da la traza es el punto de partida y el final del viaje, y no los puntos por los que va pasando el taxi. Esto quiere decir que, el *plugin* nos une los puntos gracias a la API de Google Direction pero no es la ruta real ni es exactamente esa la más frecuente. Simplemente se ha buscado un ejemplo de ruta para la ruta más frecuente que viene determinada por los cuadrados de la cuadrícula y no por puntos exactos como se ha comentado anteriormente.

A continuación se muestra el resultado final de la visualización con las rutas de ejemplo por hora, medio día y día.

Figura 34. Las diez rutas más frecuentes del día 1 de Enero del 2013 a las 00:00h.

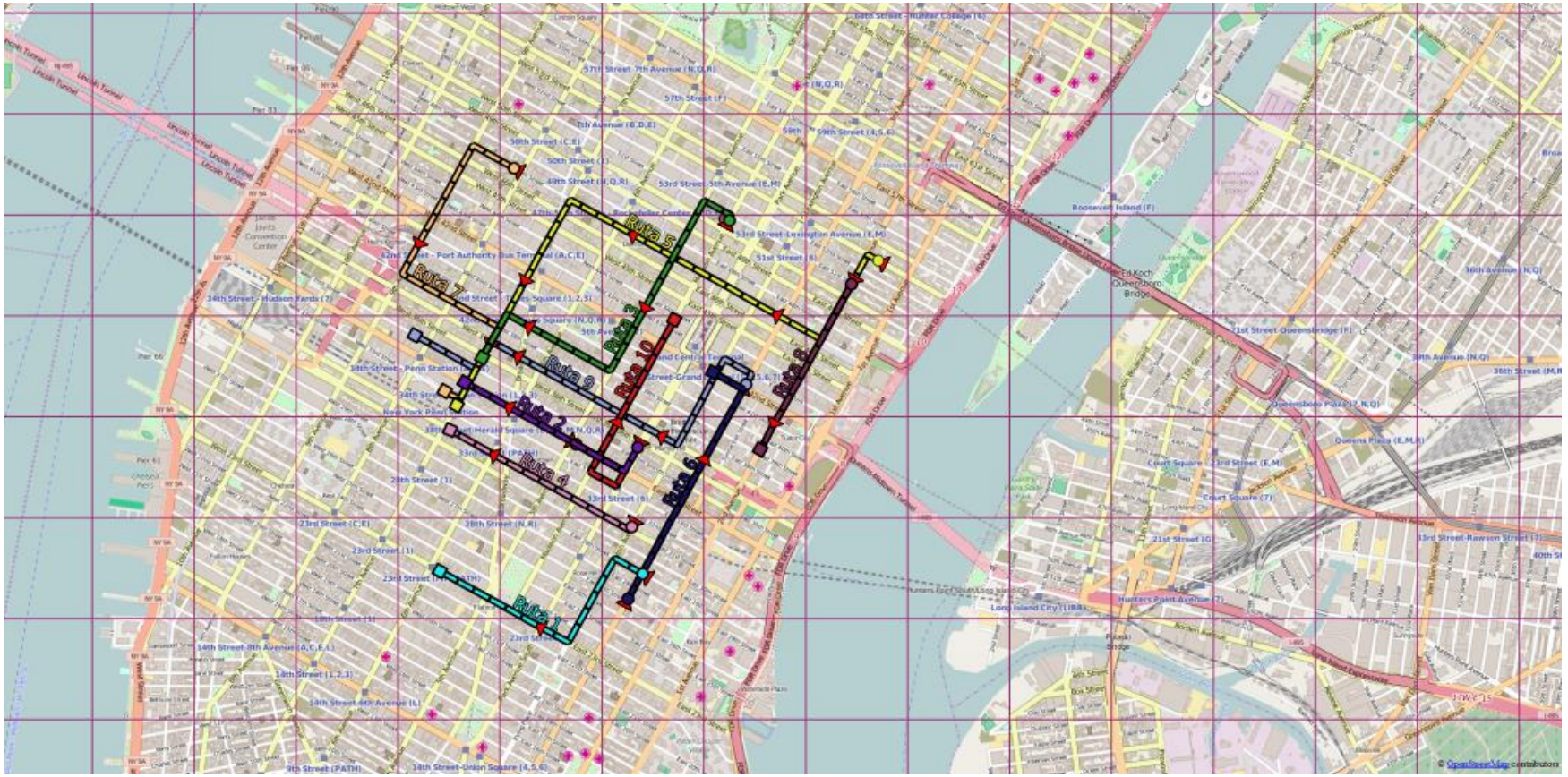


Figura 35.Las diez rutas más frecuentes del día 1 de Enero del 2013 entre las 00:00h y las 12:00h.

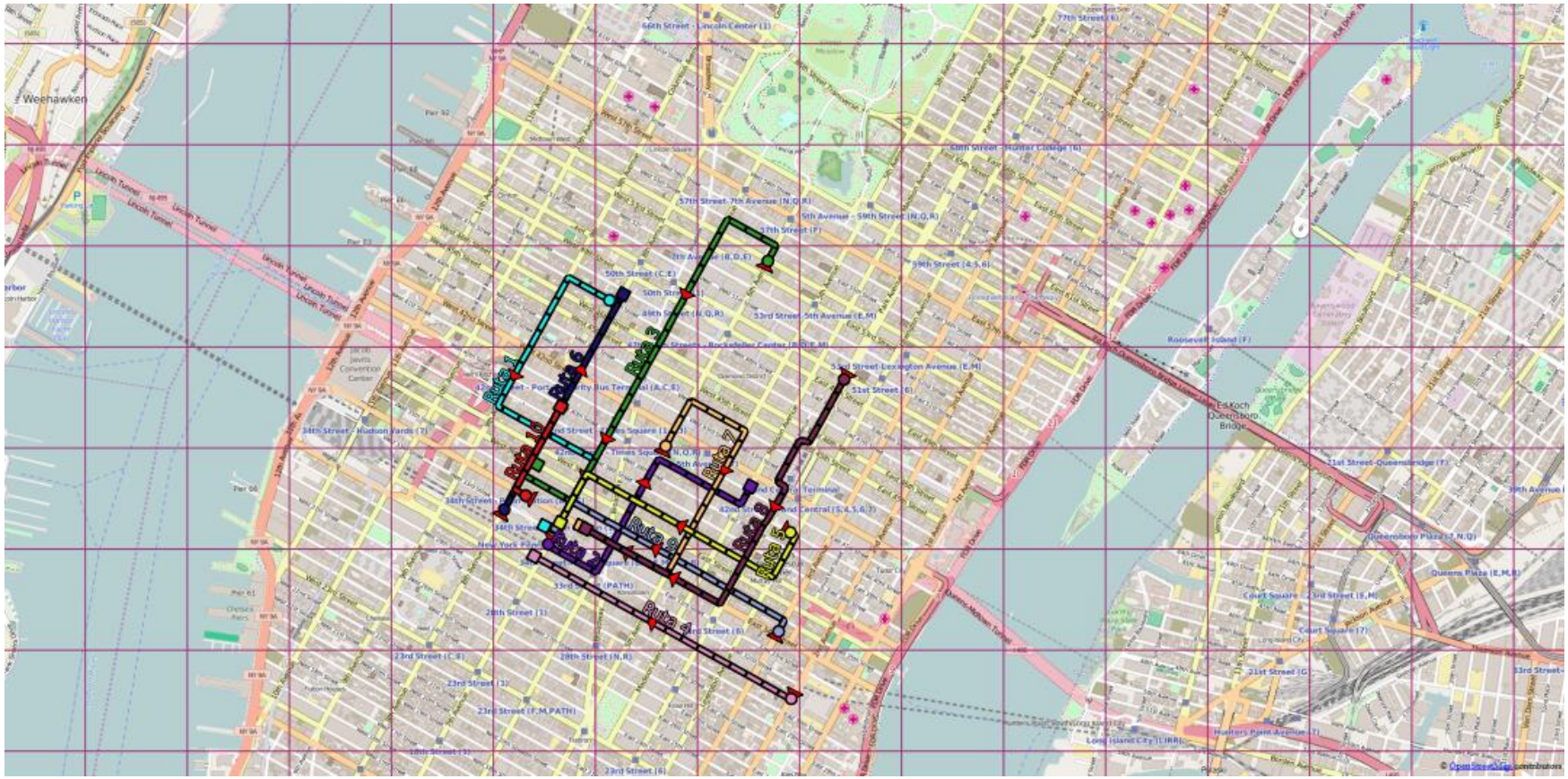


Figura 36. Las diez rutas más frecuentes del día 1 de Enero del 2013.

Capítulo 5

5. Pruebas y resultados

5.1. Introducción

Tras haber diseñado e implementado el sistema *big data*, se va a probar en diferentes entornos con el fin de detectar limitaciones y conocer la escalabilidad del sistema. Estas pruebas se hacen en base al tamaño del fichero que se va a introducir en el *almacén de datos* y a las distintas configuraciones del entorno.

Ambas pruebas son complementarias, es decir, no se realizan por separado. Para cada tamaño de fichero se probará en las diferentes configuraciones del entorno en VirtualBox. En el caso del entorno en AWS será distinto, como se detallará más adelante.

5.2. Tamaño de fichero

El objetivo de esta prueba es determinar cuál debería ser el tamaño del fichero que tendría que procesar cada nodo del cluster y como consiguiente el tiempo que tardaría en ello. Para la realización, se va a duplicar el tamaño del fichero que se calcula en función del número de trazas que contiene, no por su tamaño en bytes.

En la siguiente tabla se muestran los ficheros de prueba utilizados con su número de trazas y su tamaño.

Nombre del fichero	Número de trazas	Tamaño(Kbytes)
0_data.csv	0	0
1_data.csv	1	1
50_data.csv	50	10
100_data.csv	100	19
200_data.csv	200	38
400_data.csv	400	75
800_data.csv	800	150
1600_data.csv	1600	300
3200_data.csv	3200	600
6400_data.csv	6400	1.200
12800_data.csv	12800	2.402
25600_data.csv	25600	4.809
51200_data.csv	51200	9.620
102400_data.csv	102400	19.232
204800_data.csv	204800	38.437
409600_data.csv	4096	76.867
819200_data.csv	819200	153.694
1638400_data.csv	1638400	307.331
3276800_data.csv	3276800	614.607
6553600_data.csv	6553600	1.229.357

Tabla 3. Ficheros de prueba.

Como se ve en la Tabla 3, los tres primeros ficheros no son el doble que el anterior ya que se ha supuesto que son ficheros demasiado pequeños para que varíe su comportamiento.

5.3. Entornos de prueba

5.3.1. VirtualBox

En relación a la máquina virtual, se han hecho dos configuraciones distintas para la realización de las pruebas. Las dos configuraciones son prácticamente iguales, lo único diferente entre ellas es la memoria base.

Las características de las configuraciones 1 y 2 pueden verse en la Tabla 4 y Tabla 5 respectivamente.

General
Nombre: UbuntuDesktopVM1 Tipo: Linux Versión: Ubuntu (64 bits)
Sistema
Memoria base: 1024 MB Procesadores: 2 Orden de arranque: Disquete, Óptica, Disco duro Aceleración: VT-x/ADM-V, Paginación anidada, PAE/NX
Pantalla
Memoria de video: 12MB Servidor de escritorio remoto: Inhabilitado Captura de vídeo: Inhabilitado
Almacenamiento
Controlador: IDE IDE secundario maestro: [Unidad óptica] Vacío Controlador: SATA Puerto SATA 0: UbuntuDesktopVM1.vdi (Normal, 40,00GB)
Audio
Controlador de anfitrión: Windows DirectSound Controlador: ICH AC97
Red
Adaptador 1: Intel PRO/1000 MT Desktop (NAT)
USB
Controlador USB: OHCI Filtros de dispositivos: 0(0 activo)
Carpetas compartidas
Ninguno
Descripción
Ninguno

Tabla 4. Configuración 1 de la máquina virtual.

General
Nombre: UbuntuDesktopVM1 Tipo: Linux Versión: Ubuntu (64 bits)
Sistema
Memoria base: 1024 MB Procesadores: 2 Orden de arranque: Disquete, Óptica, Disco duro Aceleración: VT-x/ADM-V, Paginación anidada, PAE/NX
Pantalla
Memoria de video: 12MB Servidor de escritorio remoto: Inhabilitado Captura de vídeo: Inhabilitado
Almacenamiento

Controlador: IDE IDE secundario maestro: [Unidad óptica] Vacío Controlador: SATA Puerto SATA 0: UbuntuDesktopVM1.vdi (Normal, 40,00GB)
Audio
Controlador de anfitrión: Windows DirectSound Controlador: ICH AC97
Red
Adaptador 1: Intel PRO/1000 MT Desktop (NAT)
USB
Controlador USB: OHCI Filtros de dispositivos: 0(0 activo)
Carpetas compartidas
Ninguno
Descripción
Ninguno

Tabla 5. Configuración 2 de la máquina virtual.

5.3.2. AWS

Por motivos de costes en la utilización de Amazon Web Service (véase 2.4) se han reducido las pruebas en la *nube*.

La instancia EC2 elegida para los nodos del cluster tiene las siguientes características:

- API Name: c1.xlarge
- Memoria RAM: 7 GB
- Cores: 8
- Almacenamiento: 1680 GB (4 * 420 GB)
- Tipo de sistema operativo: 64 bits
- Rendimiento de red: Alta
- Coste bajo demanda: 0.84 $/hora

Y los *clusters* que se han creado para realizar las pruebas son:

- 1 Nodo: 1 maestro (c1.xlarge)
- 2 Nodos: 1 maestro (c1.xlarge) y 1 esclavo (c1.xlarge)
- 4 Nodos: 1 maestro (c1.xlarge) y 3 esclavos (c1.xlarge)
- 8 Nodos: 1 maestro (c1.xlarge) y 7 esclavos (c1.xlarge)

5.4. Resultados

Se recuerda que el objetivo de las pruebas es ver las limitaciones del sistema en función de las configuraciones del entorno. En VirtualBox se han realizado las pruebas para todos los fichero de la Tabla 3 y se ha decidido escoger el fichero de 3276800 registros para el entorno en AWS por reducción de costes, ya que es un servicio de pago bajo demanda.

A continuación, se van a mostrar los tiempos de ejecución y la velocidad en el procesamiento de los datos. Primero se explicarán los resultados de las configuraciones en la máquina virtual, y después los resultados en AWS.

En la Figura 37, se muestran los tiempos y velocidades para cada fichero en la máquina virtual de 1GB de RAM. Se observa que hasta los 204.800 registros procesados el tiempo es prácticamente el mismo. Esto quiere decir, que el sistema no es rentable hasta ese tamaño de fichero, ya que el coste en tiempo de procesar 0 datos o 204.800 es casi igual. Sin embargo, cuando procesamos el doble (405.600 registros) existe un comportamiento anómalo, ya que baja la curva de tiempo pero a partir de entonces el tiempo empieza a aumentar a duplicarse para los siguientes ficheros. Por esta razón, desde los 405.600 registros el tamaño de fichero y el tiempo de ejecución crecen casi en la misma proporción.

Al hilo de esta última conclusión, se explica el comportamiento de la velocidad, que aumenta hasta los 204.800 registros y parece mantenerse estable cuando el tiempo de ejecución es casi el doble para cada fichero.

La razón de no tener resultados para más ficheros de datos es que para el doble del último (13.107.200) el proceso no llega a finalizarse.

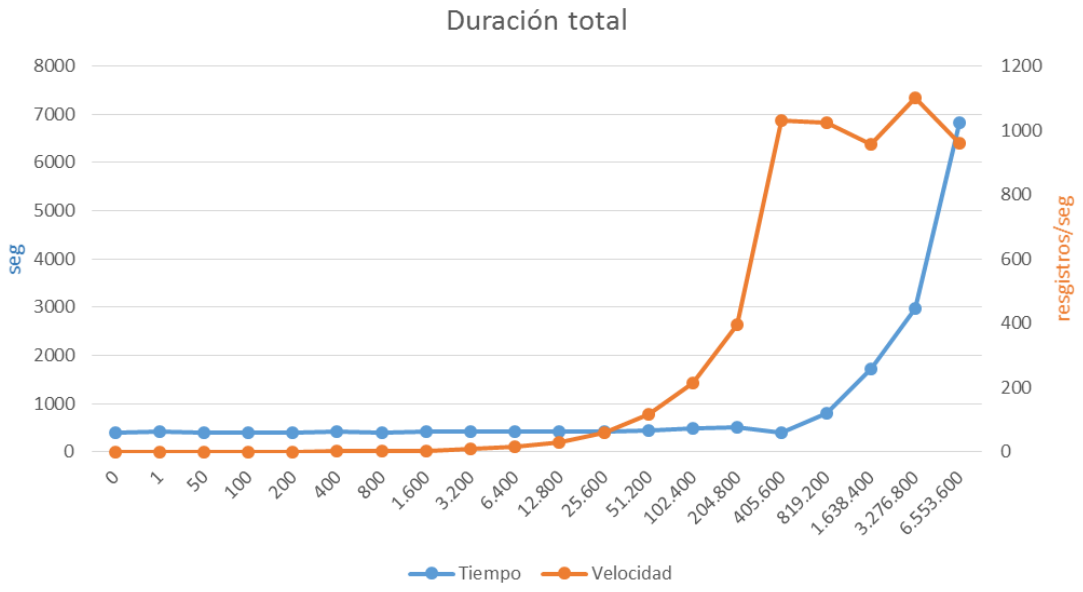


Figura 37. Tiempos y velocidad en la conf. 1 de VirtualBox para el proceso completo.

Para la segunda configuración de la máquina virtual (véase Tabla 5), se tiene un comportamiento similar al anterior. En su correspondiente gráfica (véase Figura 38) se puede observar que la velocidad parece estar desplazada hacia la derecha respecto a la de la Figura 37. La línea de tiempos permanece estable hasta las 204.800 trazas. Además, presenta la misma anomalía anteriormente comentada en los 405.600 registros, apreciándose una bajada a partir de la cual empieza su crecimiento. A diferencia de la gráfica de la Figura 37, el aumento del tamaño del fichero y del tiempo de ejecución no es proporcional, ya que se observa que los tiempos no son en torno al doble del anterior. Esto se debe a que el aumento de la memoria RAM hace que pueda procesar mayor cantidad de registros en menos tiempo.

En la velocidad cabe destacar el rápido aumento que sufre la curva entre los 204.800 y 405.600 registros en ambas gráficas. Sin embargo, para esta configuración del entorno no hay estabilidad, si no que crece hasta la prueba realizada con el último fichero.

Por la misma razón que anteriormente, no existen datos para ficheros de mayor tamaño.

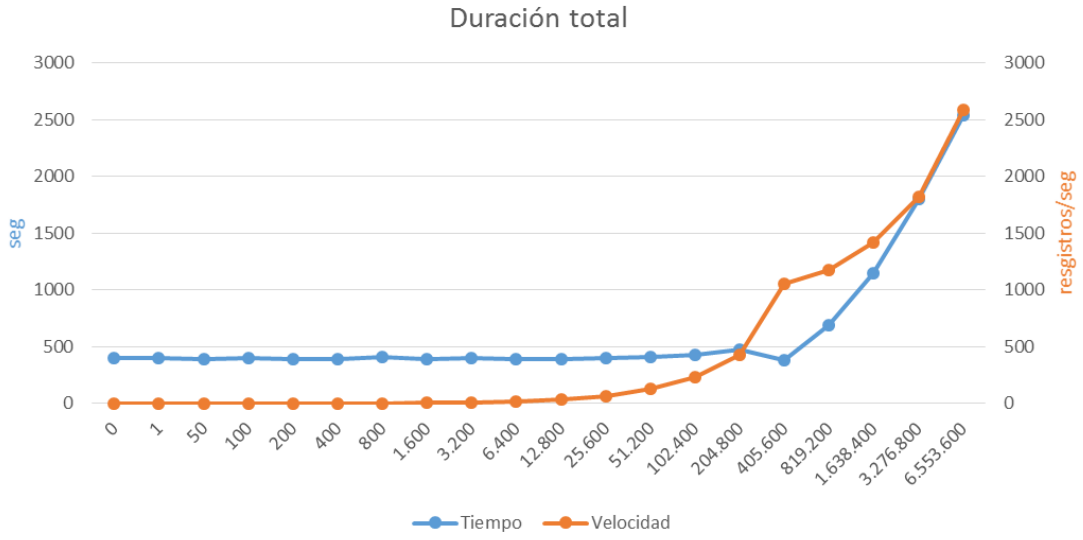


Figura 38. Tiempos y velocidad en la conf. 2 de VirtualBox para el proceso completo.

Como conclusión de ambas gráficas, podemos decir que la memoria RAM del entorno afecta directamente a los tiempos de ejecución. Pero no relaciona el tiempo y la RAM de forma proporcional. Además, otra de las conclusiones es que para el entorno de pruebas en VirtualBox el máximo fichero que se puede procesar esta entre los 6.553.600 y 13.107.200 de registros, no habiéndose podido procesar este último.

Dado que una de las conclusiones anteriores es la no linealidad del tiempo de ejecución en función de la RAM del entorno, no podemos comparar el entorno de AWS con el de VirtualBox. Sin embargo, podemos reflexionar sobre el crecimiento de la línea de tiempos. Dado que las máquinas del entorno en *cloud* son de 7GB de RAM, nos lleva a pensar que crecerá tan despacio que será prácticamente constante. Por esta razón los

tiempos de ejecución para el fichero elegido como prueba (3.276.800 de trazas) será parecido a los del intervalo entre 0 y 405.600 de la Figura 37 y la Figura 38.

El estudio que se va a realizar es como afecta el número de nodos respecto a tiempo. Tal y como vemos en la Figura 39, se mantiene el supuesto comentado anteriormente sobre el tiempo de ejecución. Se aprecia que es muy parecido al del rango en el que se mantiene en el entorno de VirtualBox.

En cuanto a la relación entre el tiempo y el número de nodos (véase Figura 39) se puede ver un aumento en la medida de tiempo para 2 nodos. Esto resulta contradictorio al razonamiento lógico de "mayor número de nodos, menor tiempo de respuesta" [47], pero tiene su explicación. De 1 a 2 nodos el tiempo de ejecución aumenta por el envío de datos entre los nodos del *cluster*. Sin embargo, cuando incrementamos los nodos a 4 u 8 el tiempo disminuye. Esto se debe a que la potencia de cálculo que se le añade al sistema compensa con coste del tiempo en el envío de los datos entre los nodos del *cluster*. El comportamiento en el aumento de tiempo al pasar de 1 a 2 nodos se asemeja a uno de los casos estudiados en el artículo [51].

La explicación al comportamiento de la curva de velocidad queda evidenciada con el razonamiento anterior.

La conclusión extraída es que para determinar el número de nodos que debe tener el *cluster* del sistema se tiene que buscar un equilibrio entre la latencia introducida por la comunicación entre los nodos y la capacidad de computo añadida. Es importante tener en cuenta que habrá un punto de optimización en relación al número de nodos a partir del cual el tiempo de ejecución no disminuya más o incluso crezca por la misma razón comentada anteriormente en el caso de 2 nodos.

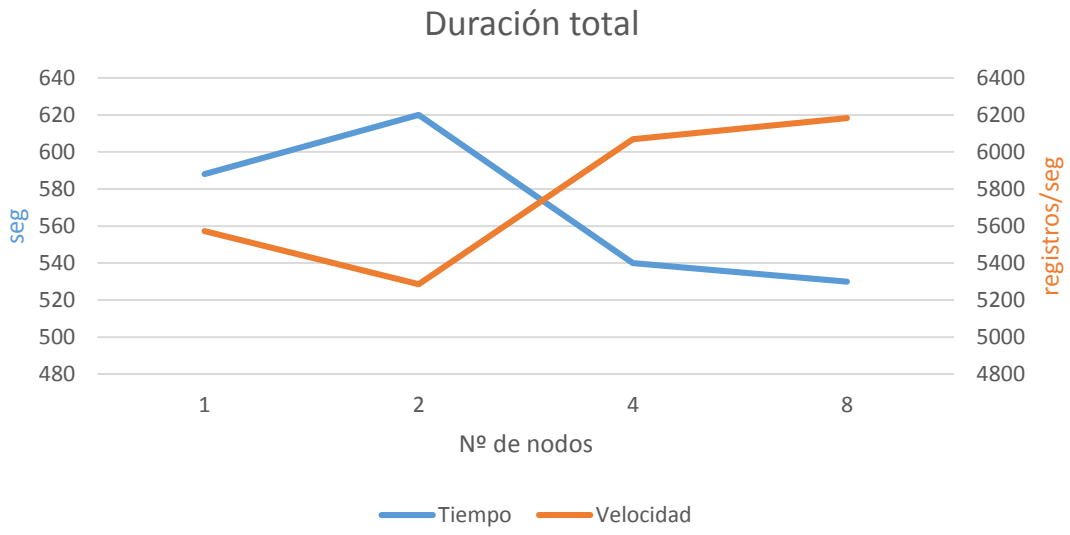


Figura 39. Tiempos y velocidad en AWS para el proceso completo.

Bloque IV

Conclusiones y futuras líneas de trabajo

Capítulo 6

6. Conclusiones y futuras líneas de trabajo

6.1. Conclusiones

El objetivo de este Trabajo Fin de Grado era el desarrollo de un sistema *big data* que ofreciera la posibilidad al usuario de interactuar con el mismo pudiendo llevar a cabo la carga, el procesado y la visualización de los datos. En este caso, las trazas correspondían a viajes de taxis en Nueva York, y el resultado final es la representación de las diez rutas más frecuentes por horas en cada día, en los medios días de cada día y por día.

Después de haber realizado el diseño y la implementación del sistema se puede concluir con el cumplimiento de los requisitos especificados. El usuario tiene la capacidad de cargar los datos en HDFS con la facilidad de uso que proporciona la comunicación a través de la consola de comandos. También realiza el procesamiento de los datos aislándose de la complejidad que conlleva la programación del *Mapreduce* gracias a la utilización de la tecnología *Hive.* Y por último, cabe destacar la principal ventaja de utilización de QGIS, que es ser una herramienta de código libre y disponibilidad gratuita.

6.2. Futuras líneas de trabajo

Algunas de las ideas como trabajo a futuro para la ampliación de funcionalidades del sistema o como pruebas del mismo son:

- Visualización de estadísticas extraídas en el procesamiento de los datos. En este caso, al tratarse de rutas de taxis puede ser el porcentaje de clientes que utilizan cada medio de pago (efectivo o tarjeta), la utilización de taxis por horas o las distancias medias recorridas a lo largo del día.
- Extracción de patrones de comportamiento en función de parámetros externos, es decir, combinación de datos de las rutas de taxis con datos climatológicos o datos de eventos sociales.
- Utilización de trazas con distinta naturaleza, no relacionados con el tráfico.

Bloque V

Planificación y presupuesto

Capítulo 7

7. Planificación

7.1. Fases de desarrollo

La duración de la realización del proyecto ha sido alrededor de seis meses. A continuación se resumen las fases en las que se ha dividido el desarrollo y el tiempo empleado para cada una de ellas:

1. Planteamiento de la necesidad y descripción del problema

   En esta fase se pretende centrar la idea general del proyecto. Se plantea la necesidad de probar nuevas tecnologías y se acuerda cuáles se van a utilizar en concreto para el desarrollo.
   - Tiempo estimado: 5 días.
   - Participantes: Tutor y desarrollador del proyecto.

2. Estudio de las tecnologías utilizadas

   Análisis de las tecnologías que se van a emplear. Se hizo un estudio de *Hadoop* y *Hive*, que son las principales en el desarrollo del proyecto. También se estudia la alternativa del entorno en *cloud*, en concreto AWS.
   - Tiempo estimado: 20 días.
   - Participantes: Tutor y desarrollador del proyecto.

3. Diseño del sistema

   Fase en la que se definen los requisitos del sistema y la arquitectura de componentes que lo forman, así como la interacción entre ellos y la elección del diseño final.
   - Tiempo: 15 días.
   - Participantes: Tutor y desarrollador del proyecto.

4. Configuración del entorno de implementación

   Instalación y configuración de las herramientas a utilizar: VirtualBox con Linux Ubuntu (64-bits), y dentro de ésta Hadoop y Hive.
   - Tiempo: 10 días.
   - Participantes: Tutor y desarrollador del proyecto.

5. Implementación

   Escritura del código necesario para el correcto funcionamiento del sistema para los requisitos establecidos.
   - Tiempo: 60 días.
   - Participantes: Tutor y desarrollador del proyecto.

6. Configuración del entorno de prueba

   Realización de cambios en la configuración del entorno de trabajo para la realización de las pruebas pertinentes. Además, se configura el entorno en *cloud.*
   - Tiempo: 5 días.
   - Participantes: Tutor y desarrollador del proyecto.

7. Pruebas y conclusiones

   Pruebas de validación del correcto funcionamiento del sistema en diferentes entornos. También se miden tiempos de ejecución y se contrastan los resultados.
   - Tiempo: 30 días.
   - Participantes: Tutor y desarrollador del proyecto.

8. Redacción de la memoria

   Documentación final de todo el trabajo realizado en el desarrollo completo del proyecto.
   - Tiempo: 45 días.
   - Participantes: Tutor y desarrollador del proyecto.

Además de las fases indicadas anteriormente, a lo largo del desarrollo del proyecto se ha establecido una fase intercalada entre todas las demás de seguimiento del trabajo que comprende una serie de reuniones con el tutor para aprobar y verificar los avances del desarrollo (esta fase no será contemplada en el diagrama de Gantt).

7.2. Diagrama de fases de ejecución

En el diagrama de Gantt (véase Figura 31), se representa el tiempo que se ha dedicado a cada una de las fases del proyecto indicadas anteriormente.

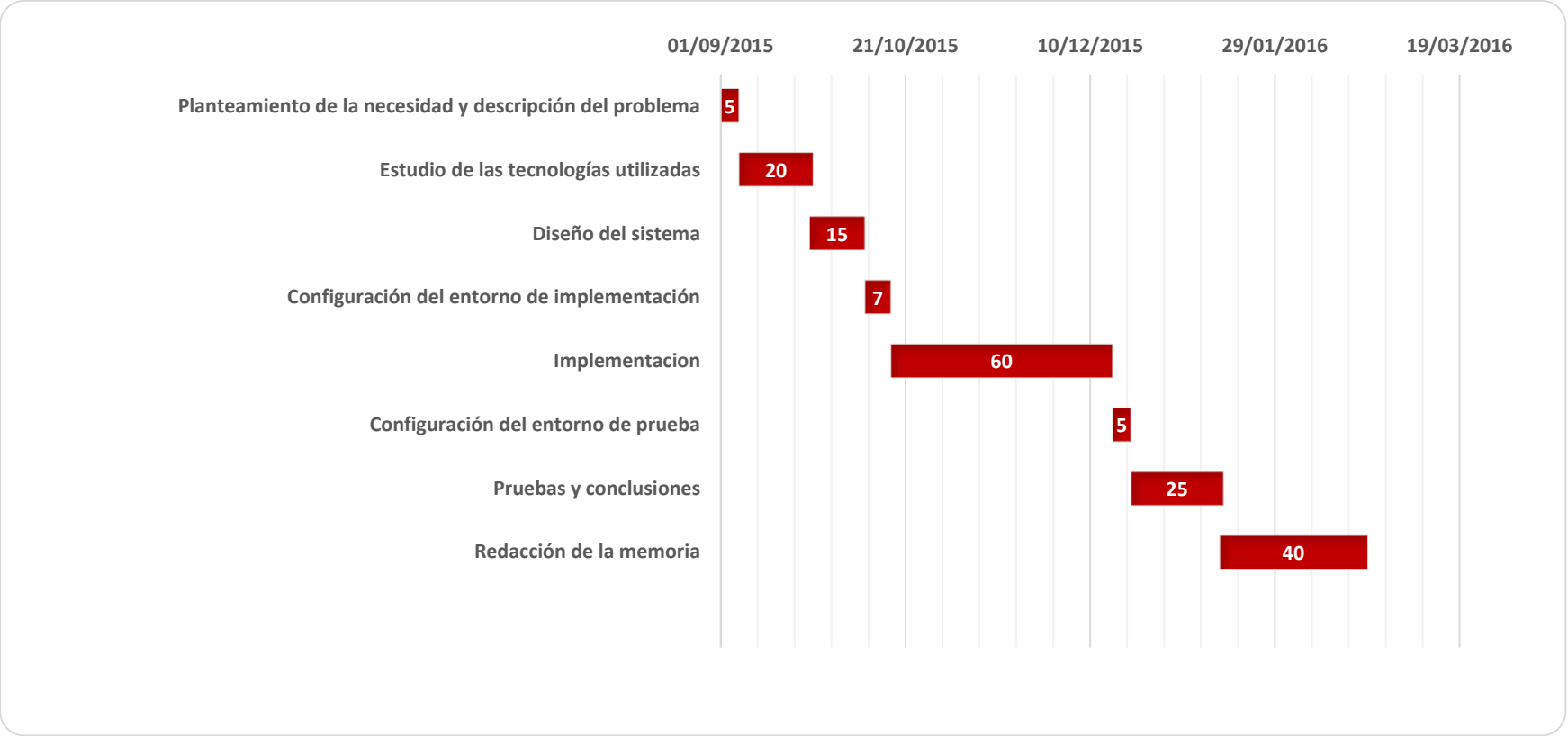


Figura 40.Diagrama de Gantt.

Capítulo 8

8. Presupuesto

8.1. Medios empleados

Los recursos empleados en el desarrollo del proyecto son los siguientes:

Recursos humanos

- 1 Ingeniero Junior
- 1 Ingeniero Senior

Recursos materiales

- 1 PC portátil de gama media
- Microsoft Office 2013

Otros recursos

- Conexión a Internet durante 6 meses
- Servicios de AWS
    - Instancia EC2 c1.xlarge
    - Almacenamiento S3

Software open source (gratuito)

- VirtualBox versión 5.0.10
- Apache Hadoop versión 2.6.0
- Apache Hive versión 0.13.0
- QGIS versión 2.12.2

8.2. Presupuesto del trabajo

1. Autor: Montserrat Murillo González
2. Departamento: Ingeniería Telemática
3. Descripción del proyecto
    - Título: Sistema *big data* para el análisis de rutas de taxis en NYC
    - Duración: 6 meses
    - Tasa de costes indirectos: 20%
4. Presupuesto total del proyecto 21181,45€
5. Desglose del presupuesto, costes directos:

COSTES MATERIALES					
Concepto	**Cantidad**	**Coste (€)**	**Dedicación (meses)**	**Periodo de depreciación (meses)**	**Coste imputable (€)**
PC portátil	1	500,00	6	48	62,5
Microsoft Office 2013	1	119,00	6	48	14,88
				TOTAL	77,38

Tabla 6. Costes materiales del proyecto.

En la realización del proyecto se han visto involucradas dos personas: 1 ingeniero junior con una dedicación de media jornada y un ingeniero senior con dedicación del 10% de la jornada. En la tabla 7 se recogen los tiempos dedicados por cada una de estas personas para cada tarea del proyecto.

Tarea	**Dedicación total (días)**	**Dedicación del Ingeniero Junior (horas)**	**Dedicación del Ingeniero Senior(horas)**
Planteamiento de la necesidad y descripción del problema	5	20	4
Estudio de las tecnologías utilizadas	20	80	16
Diseño del sistema	15	60	12
Configuración del entorno de implementación	7	28	5,6
Implementación	60	240	48
Configuración del entorno de prueba	5	20	4
Pruebas y conclusiones	25	100	20
Redacción de la memoria	40	160	32
	TOTAL	708	141,6

Tabla 7. Resumen de tareas y tiempo dedicado del ingeniero junior y senior.

COSTES DE PERSONAL				
Concepto	**Cantidad**	**Dedicación (horas)**	**Coste por hora (€)**	**Coste total (€)**
Ingeniero Senior	1	141,6	30	4248
Ingeniero Junior	1	708	18	12744
			TOTAL	16992

Tabla 8. Costes de personal del proyecto.

OTROS COSTES				
Concepto	**Cantidad**	**Dedicación (horas)**	**Coste por hora (€)**	**Coste total (€)**
Conexión a Internet	1	180	1,16	208,8
Instancia EC2	1	20	0,462	9,24
Almacenamiento S3	-	-	-	0,026*
			TOTAL	218,07

Tabla 9. Otros costes del proyecto.

* El precio del almacenamiento en S3 es del primer TB/mes (las pruebas se realizaron dentro del mismo mes) y el coste total es de 0,026€/GB (el fichero de prueba es de 600MB aproximadamente).

COSTE TOTAL	
Concepto	**Coste (€)**
Material	77,38
Personal	16992
Otros	218,07
TOTAL	17287,45

Tabla 10. Coste total del proyecto.

Bloque VI

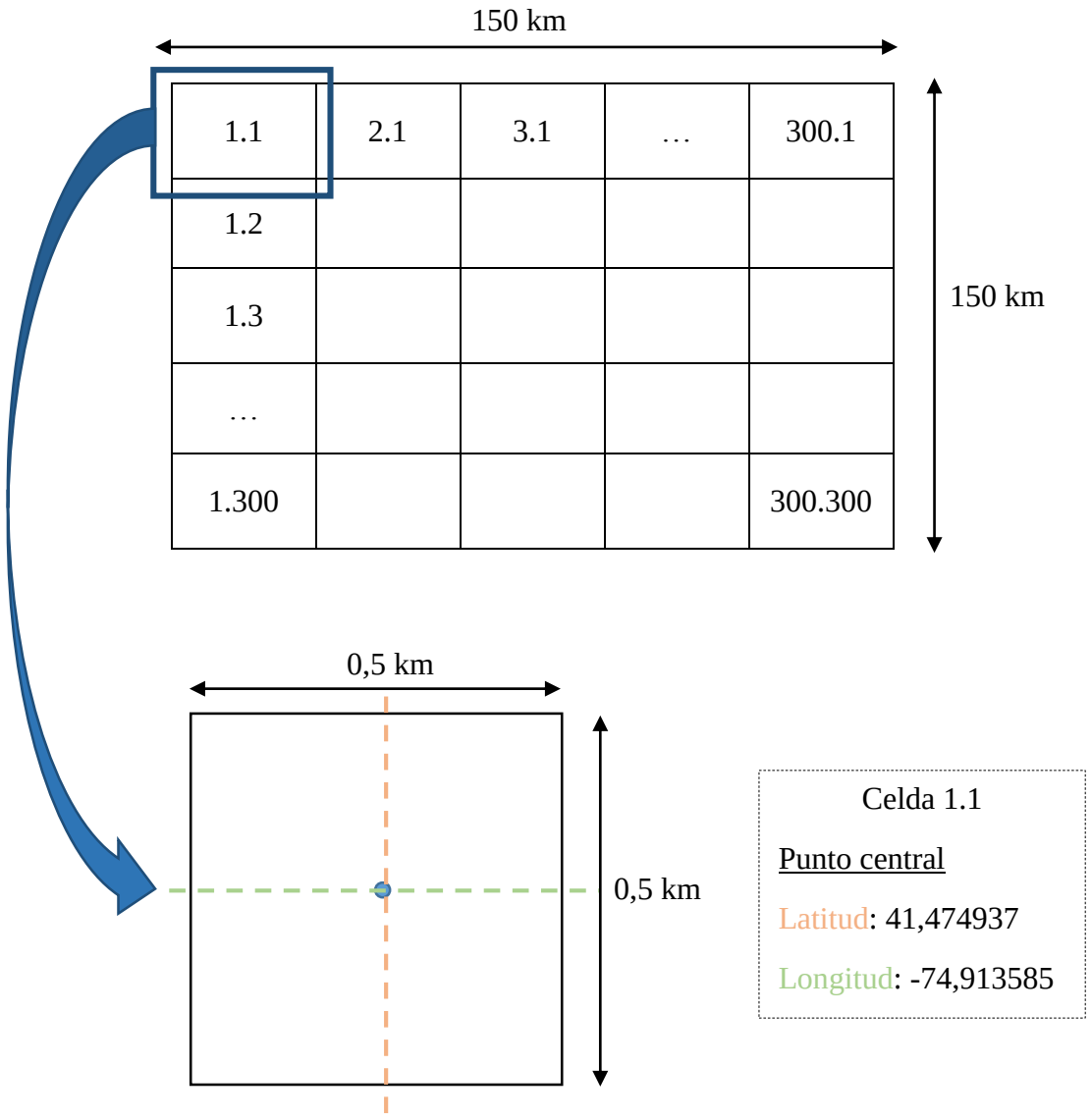
Anexos

Anexo A: Cuadrícula para la visualización de rutas

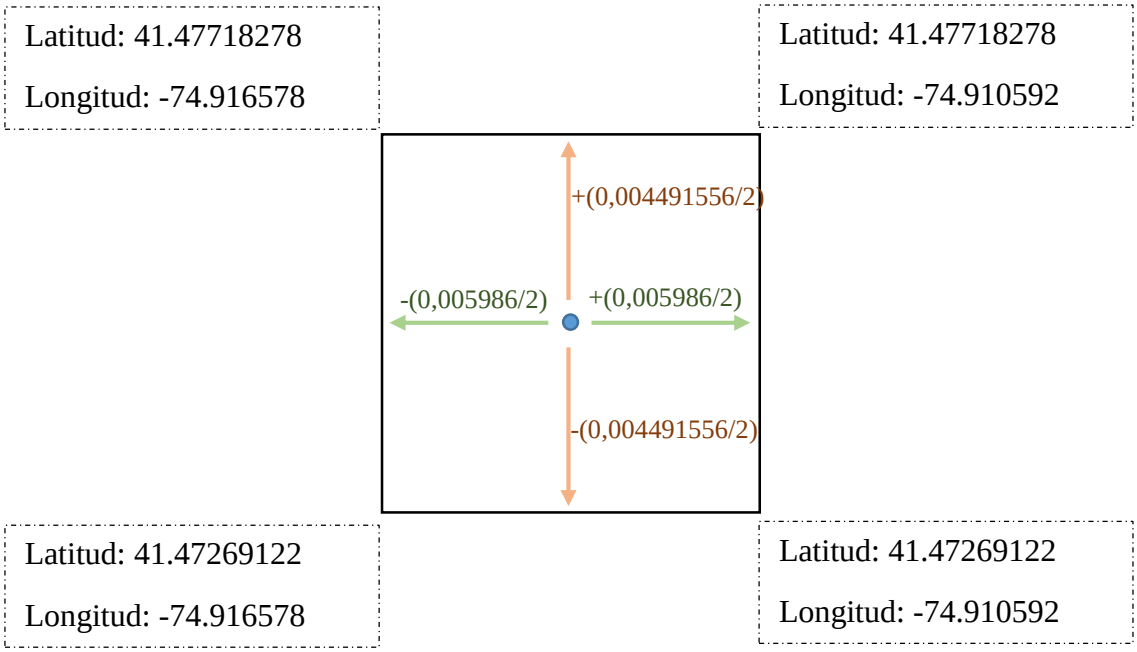
Características de la cuadrícula

Algunas de las características de la cuadricula que se deben tener en cuenta a la hora de la representación son:

- Es de 300 x 300.
- Cada cuadrado de la cuadrícula corresponde a 0,5 km. Esto corresponde a 0,004491556 en latitud y 0,005986 en longitud.
- El único punto inicial que se tiene es el central del cuadrado 1.1, siendo de 41,474937 de latitud y -74,913585 de longitud.



Coordenadas de las esquinas de la primera celda



Se guardaran las coordenadas de todos los cuadrados de la cuadricula en un .txt calculados mediante el código de Python (véase Figura 41). El formato de los registros será: número de cuadrado longitud mínima longitud máxima latitud mínima latitud máxima. Por ejemplo, para el primer cuadrado el registro es:

1.1 -74.916578 -74.910592 41.47269122 41.47718278

```python
pi=[1.1,-74.916578,-74.910592,41.47269122,41.47718278]
cuadric=[pi]
cuadricula=[]
listaCoordenadas=[]
puntoLista=''

for h in range(300):
    if h==0:
        # Se añade el punto inicial en la lista antes de calcular el resto de puntos
        puntoLista=str(pi[0]) + '\t' + str(pi[1]) + '\t' + str(pi[2]) + '\t' + str(pi[3]) + '\t' + str(pi[4]) +'\t'
        listaCoordenadas.append(puntoLista)

        # Se calculan los puntos de la primera fila de la cuadricula,excepto el primero que es el inicial
        # El bucle va desde 0 a 298 (la iteracion de 299 no se hace) asi tendriamos incluyendo el 0, 299
        # punto mas el punto inicial 300.
        for i in range(0,299):

            pnuev=[str(i+2)+'.1',pi[2],pi[2]+0.005986,pi[3],pi[4]]
            cuadric.append(pnuev)
            pi=pnuev
            puntoLista=str(pnuev[0]) + '\t' + str(pnuev[1]) + '\t' + str(pnuev[2]) + '\t' + str(pnuev[3]) + '\t' + str(pnuev[4])
            listaCoordenadas.append(puntoLista)
    else:

        # Se calculan las coordenadas de todas las demás filas en función de la primera.
        for j in range(0,300):

            pnuev=[str(j+1)+'.'+str(h+1),cuadricula[h-1][j][1],cuadricula[h-1][j][2],cuadricula[h-1][j][3]-0.004491556,cuadricula[h-1][j][3]]
            cuadric.append(pnuev)
            puntoLista=str(pnuev[0]) + '\t' + str(pnuev[1]) + '\t' + str(pnuev[2]) + '\t' + str(pnuev[3]) + '\t' + str(pnuev[4])
            listaCoordenadas.append(puntoLista)

    cuadricula.append(cuadric)
    cuadric=[]

# Se gurdan las coordenadas en un archivo .txt
rutaSave=ruta='C:/Users/montse/Desktop/TFG/Ficheros/coordenadas.txt'
with open(rutaSave, 'wb') as f:
    for line in listaCoordenadas:
        f.write(bytes(line, 'UTF-8'))
        f.write(bytes('\r\n', 'UTF-8'))
```

Figura 41. Código python para el cálculo de las coordenadas de la cuadrícula.

Anexo B: Gráficas de tiempos de ejecución y velocidad

Se van a mostrar para los diferentes entornos de trabajo, las gráficas de los tiempos de ejecución y velocidad para cada parte del tratado de los datos: carga de datos, procesamiento de los datos y cálculo de las rutas. Es importante destacar que para en el gráfico de rutas se muestra el tiempo de ejecución pero sólo para las 10 rutas más frecuentes por hora de cada día, no para los tres periodos de tiempo del desarrollo del proyecto.

VirtualBox

Configuración 1

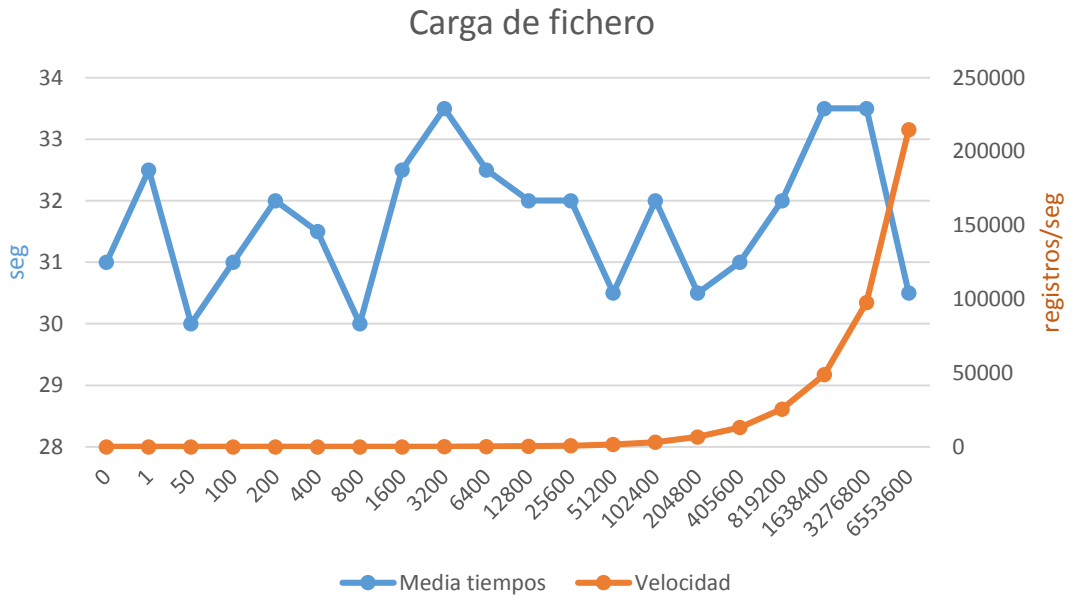


Figura 42. Tiempos y velocidad en la conf. 1 de VirtualBox para la carga del fichero.

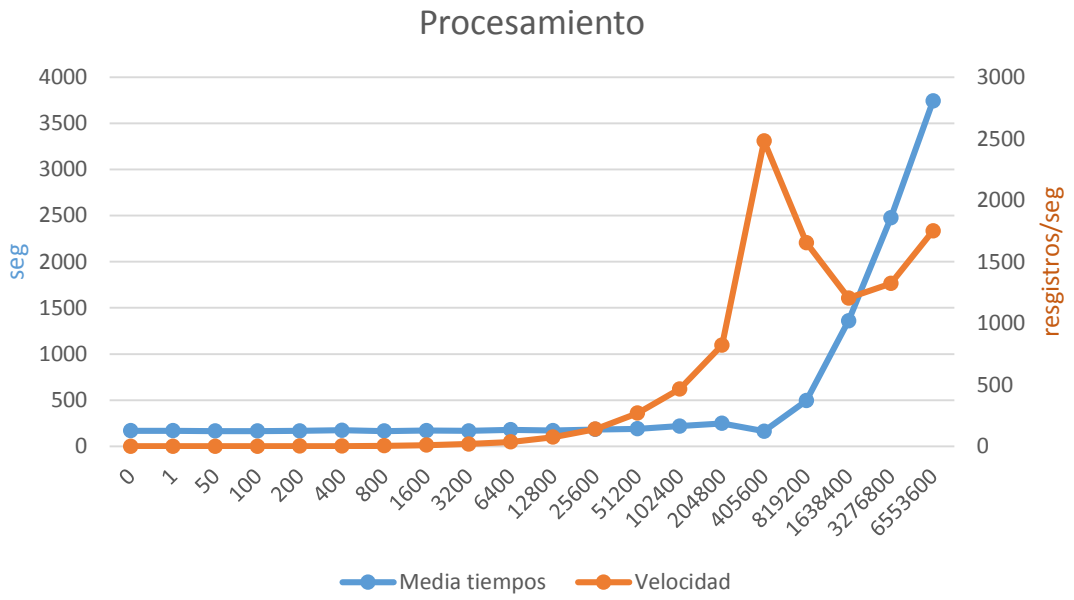


Figura 43. Tiempos y velocidad en la conf. 1 de VirtualBox para el procesamiento.

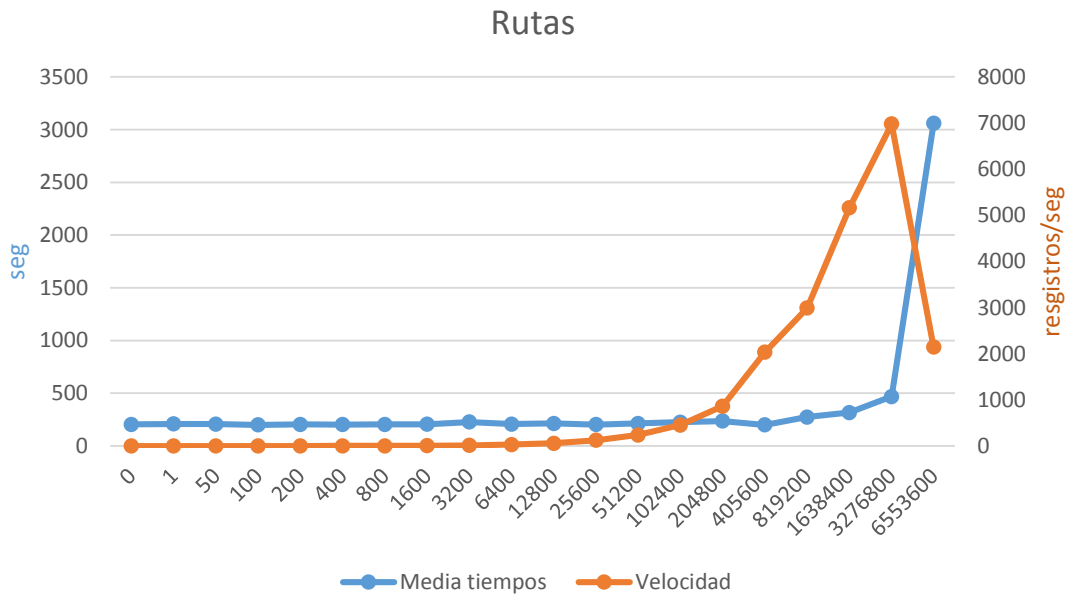


Figura 44. Tiempos y velocidad en la conf. 1 de VirtualBox para el cálculo de rutas.

Configuración 2

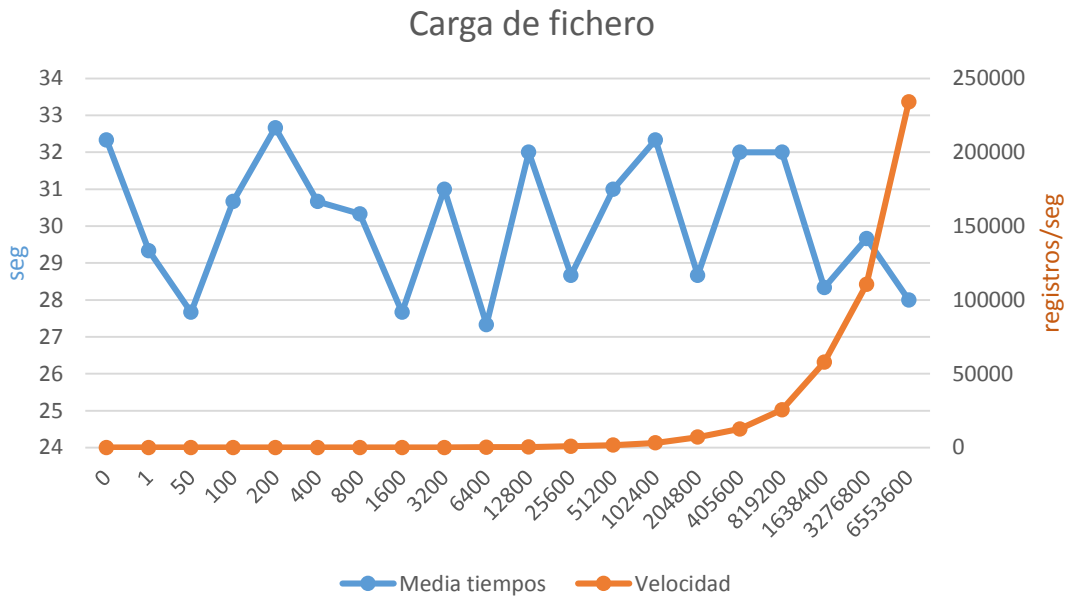


Figura 45. Tiempos y velocidad en la conf. 2 de VirtualBox para la carga del fichero.

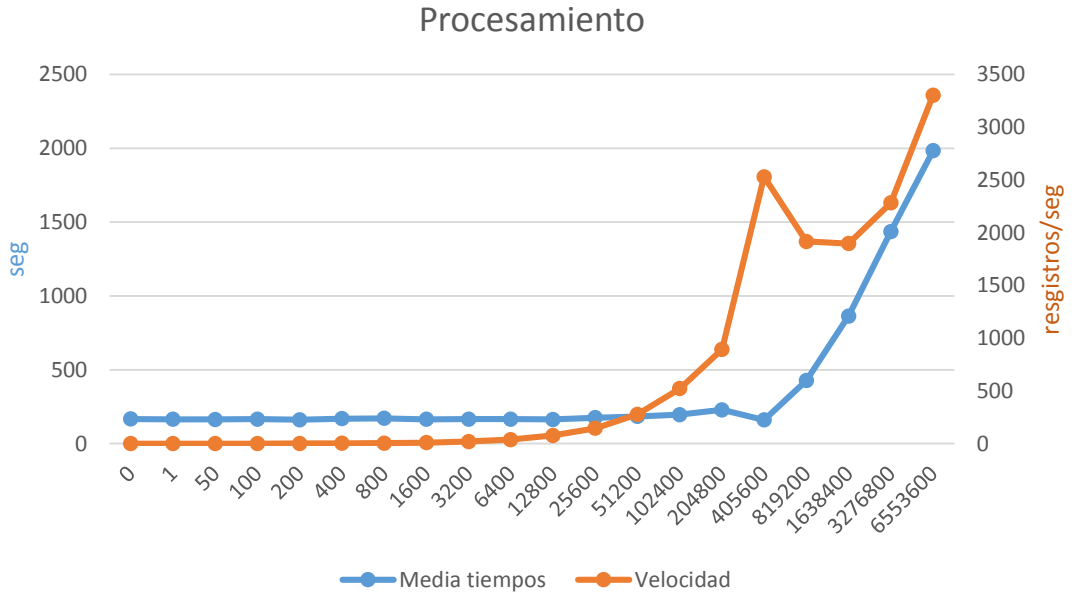


Figura 46. Tiempos y velocidad en la conf. 2 de VirtualBox para el procesamiento.

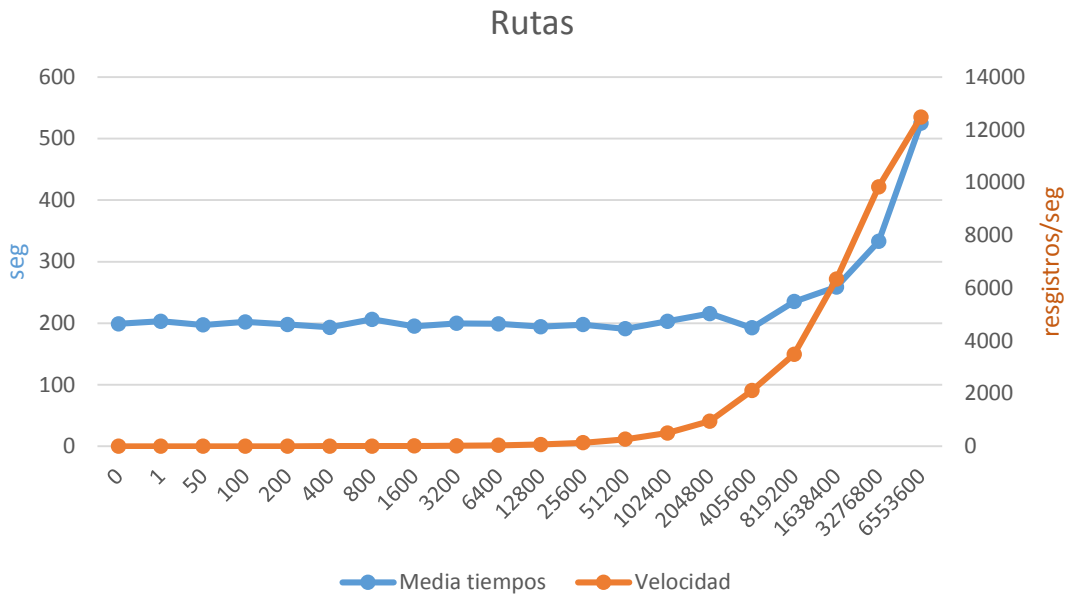


Figura 47. Tiempos y velocidad en la conf. 2 de VirtualBox para el cálculo de rutas.

Amazon Web Services

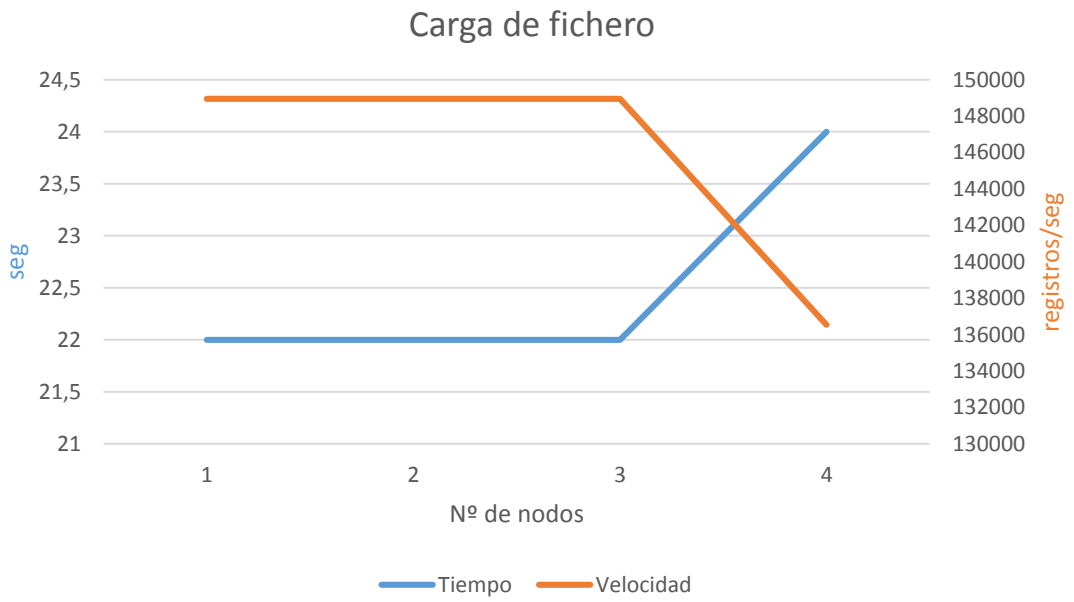


Figura 48. Tiempos y velocidad en AWS para la carga del fichero.

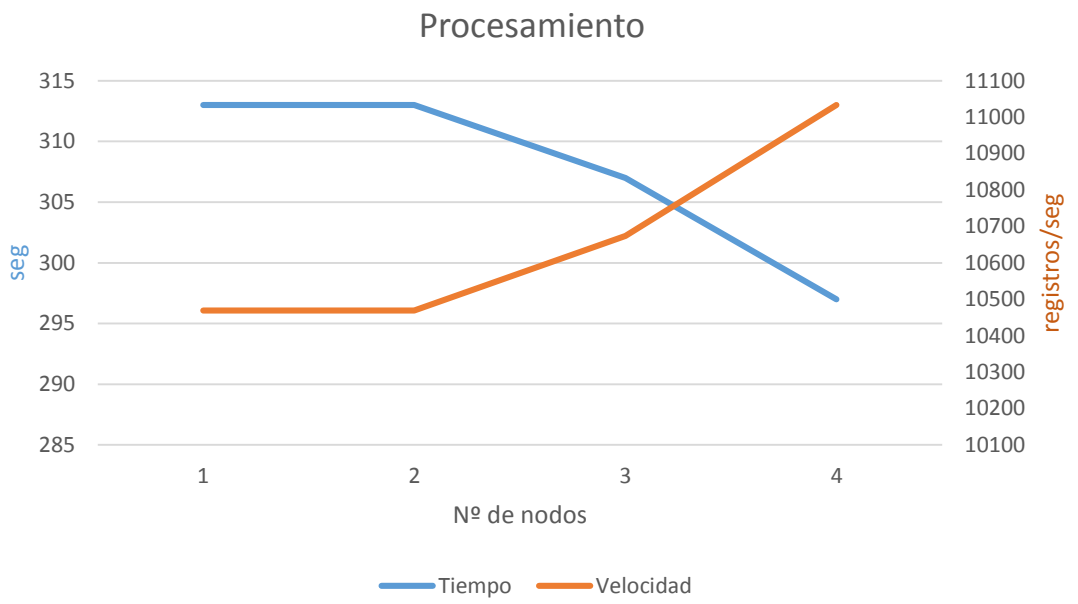


Figura 49. Tiempos y velocidad en AWS para el procesamiento.

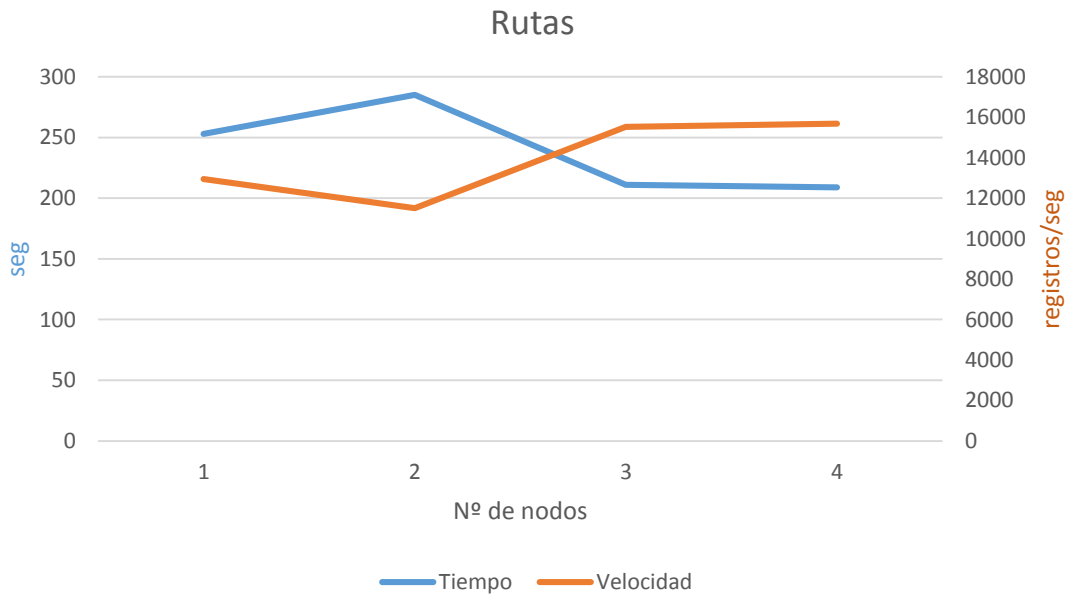


Figura 50. Tiempos y velocidad en AWS para el cálculo de rutas.

Anexo C: Extended Abstract

Context and Motivation

New technological needs arise from the advancement of information technology. Traditional tools are not able to meet those needs because there are large volumes of data [1]. This is due to increased volume of data, which make the processing of them is a costly step when looking for answers.

For this reason, organizations have had to face new challenges to obtain knowledge or information that cannot be achieved beforehand from the large set of data generated today. The use of new technologies help organizations to analyze, discover and to understand beyond what conventional tools gives them over their data, so they can increase their competence.

This project is part of the transport sector because the data used are generated by taxis in New York contributed by the DEBS 2015 Grand Challenge [2] competition. Processing of these data with technologies that support is sought. These technologies are *big data* technologies to extract information about the most common routes for hours each day, half days each day and each day. This information could help us understand the traffic generated and to make strategic decisions in the sector.

In order to obtain results, a user must be able to interact with the *bid data* system. Firstly, once the data extracted from the source they must be loaded into the system to be processed. The processing is intended both collecting valid data and reformatting them if necessary. A cleaning process is done. The wrong considered records is deleted, the fields that are to be used are kept and the fields are transformed when the target system needs it. Finally in the case of this project taxi routes on a map will be displayed.

The *big data* system is developed with the *Apache Hadoop* [3] and *Apache Hive* [4] technology using the advantages offered by distributed systems and programming abstraction techniques for processing large volumes of data. However, the system design adapts to developing other technologies and the configuration of the work environment can modify thus providing flexibility to the system.

Then the general outline of the system developed (see Figure A) in the project is illustrated. As mentioned above, the user has the ability to load data into the system, specifically in the distributed file system called Hadoop HDFS. The data can come from different origin sources, so here there is no homogeneity in the data format and structure. However, in this developed project the data are obtained from the same source. After that, using the framework Apache Hive proceed to clean the data, in other words, invalid records are discarded previously established rules for disposal and some new fields are created from existing ones. Fields not suffer any formatting process because there is no target system that needs it. As a final step, the data obtained at the output of processed data will be displayed on a map with the tool open source QGIS.

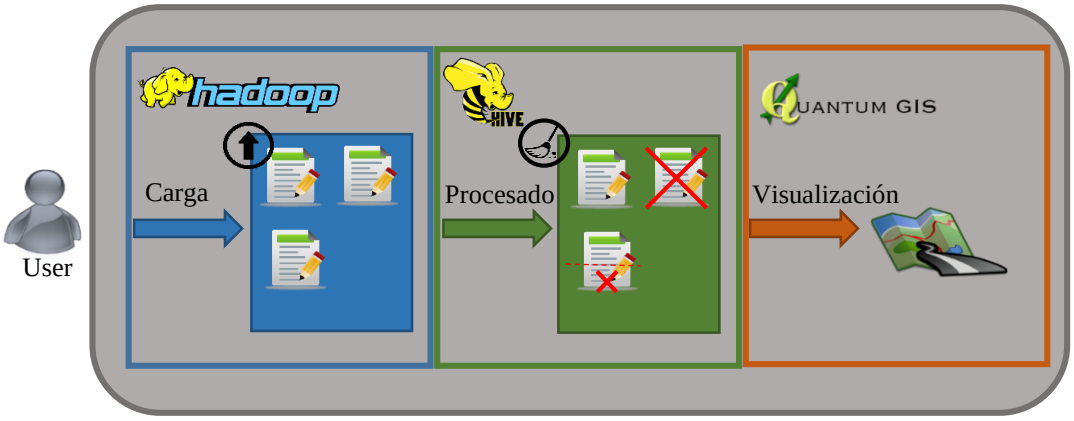


Figure A. The general outline of the system developed.

Objectives

This project involves the development of a *big data* system. The stages of development are based on the following objectives:

1. Study of the context and motivation of the project.
2. Study of the technologies that will be used in developing the system.
3. Design of a *big data* system containing all project requirements.
4. Implementation of the system based on the design.
5. Display the results of data processing.
6. Testing performance system in different environments.
7. Extracting conclusions from the project.
8. Raise future lines providing continuity to the project.

Document structure

This document is divided into parts and turn into chapters, which are listed below:

- Par one: Introduction.

    - Chapter one: Introduction and objectives.
      In this chapter the motivations of the project are presented and the objectives are set by collecting the general idea of the work being performed. In addition, the structure of memory is enunciated.

- Par two: State of the art.

    - Chapter two: State of the art.
      In Chapter 2, the technologies used in the project which should be known by the reader prior of the technical solution is explained.

- Part three: Work performed.

    - Chapter three: System design.
      This chapter describes the system design big data, as well as components that form and the interaction between them is described.
    - Chapter four: Implementation.
      Chapter 4 contains a description of the steps that have been followed in the development of the system implementation, from initial loading of data to displaying the results of processing.
    - Chapter five: Testing and results.
      This chapter describes the tests that have been conducted to know the limits of the environment and runtimes achieved are explained.

- Part four: Conclusions and future work.
    - Chapter six: Conclusions and future work.
      In Chapter 6, the analysis of the achievement of the objectives and possible future lines work are described.

- Parte five: Planning and budget.
    - Chapter seven: Planning.
      Chapter 7 sets out the stages of development of the project and its corresponding Gantt chart.
    - Chapter eight: Budget.
      This chapter describes the resources used and the budget for the project are discussed.

- Part six: Annexes.

    - Annex A: Grid for viewing routes.
    - Annex B: Charts runtimes.
    - Annex C: Extended abstract.
    - Annex D: Regulatory framework.

- References

Technologies

Big data

It is a fact that today's generation of large amounts of digital data is increasing rapidly. The company IBM [5], one of the company with great strength in the telecommunications and consulting states that the amount of data generated to date are 2.5 trillion bytes, which means that 90% of existing data in the world today have been created over the past two years [6].

Data storage is not a new practice, and less than today. The human being has done so since time immemorial. A clear example is the library, which is an information storage. From all this, the term *big data* (or macrodata) emerges as a set of tools capable of processing and analyzing large volumes of data stored on both structured and unstructured or semi-structured that cannot be treated conventionally with traditional software tools [8]. In addition, it is classified as the trend in the advancement of technology that involves a new approach to making strategic business decisions.

The big data is defined with 5 V's. Although initially were 3, the 5V's today are as follows [9]: volume (data size), velocity (speed of change), variety (different form of data source) and veracity (uncertainty of data).

The most immediate goal is the pursuit of knowledge from data in order to help the companies make right decisions even in real time, but it is also a business opportunity. Whenever the *big data* used more in business, because due to their use is a greater awareness of the needs of customers and improves the performance of products and services [8]. Even, you can get benefits from that you can feel the inner feeling of the company and being able to act within it.

Apache Hadoop

Apache Hadoop [35] is an open source framework developed in Java that allows distributed processing form of large data sets. This processing is performed on the cluster[4] nodes that comprise adding computing power and provide greater storage [36].

Data processing on a large scale requires a warehouse capable of holding large amounts of data and how to process it effectively. These two requirements of the two main components supplement core Hadoop [36]:

- HDFS (Hadoop Distributed File System): This is a distributed file system based on Google File System (GFS) and prepared for the performance takes place in

[4] Cluster: This is a set of machines with hardware components in common views together as a single computer.

large clusters. He has a master-slave architecture. There are a master (NameNode) and many slaves (DataNodes) as required.

- Hadoop MapReduce: It is a framework used to process high volumes of data in parallel on clusters. It has a first task of mapping in which a data set from a list of key-value and a second reduction task that combines the set of tuples resulting from the mapping and combining data sets key-value becomes smaller.

In addition, Hadoop consists of two more components. These components are: Hadoop common, these are Java libraries and utilities that require other modules. Hadoop Yarn, responsible for scheduling and management of all resources in the cluster (see Figure B).

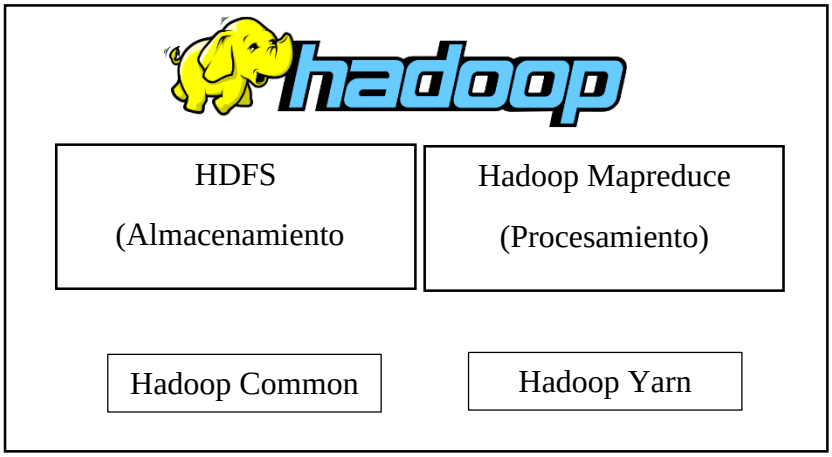


Figure B. Components of Apache Hadoop.

The main advantage of using Hadoop is the user is allowed to quickly test a distributed system because it has a high efficiency when working with several machines in parallel. It also ensures fault tolerance and high availability that they are independent of the hardware used, so allowing the introduction or removal of new nodes without disrupting the smooth operation of the cluster. Finally, one of its major advantages, besides being open source, is reflected in support for all platforms as it is developed in Java.

HDFS

HDFS is responsible for storing the data in the cluster, and unlike other distributed file systems is very fault-tolerant and it is designed with low-cost hardware [38].

The data storage is made between cluster nodes. The data are divided into blocks of 64MB and stored on multiple machines, in other words, the blocks are replicated to different nodes to ensure fault tolerance and if a node loses communication with the cluster, the data can find in other nodes [37].

The effects of data replication are: Increased system reliability because there are multiple copies of data, greater availability to meet these replicas on multiple nodes in the cluster, and also increased performance because the data is processed more effectively [37].

Therefore, HDFS characteristics [38] focus on the storage capacity and processing data provided. In addition, it offers a friendly commands interface for the user with which you can interact with HDFS and even know the current state of the cluster at all times. It also allows access to the data of continuous and uninterrupted manner and provides authentication and file permissions. However, there are disadvantages to their use as latency caused by the communication between nodes, the modifications are made at the end or there can't be multiple writers only can be a single writer [39].

Apache Hive

The origin of Hive [40] come from Facebook because it was its creator. This framework allows to work with the distributed Hadoop file system and it also facilitates data management [41].

Hive's dialect is similar to SQL called HQL, so its operation is based on consultations on large volumes of data stored in HDFS. Consultations translate into MapReduce jobs, so Hive facilities and abstracts us of complex programming tasks on mapping and data reduction [41].

Hive is not considered a database, but has aspects in common as the need to create a tables schema with its columns and their types to work with data or load this data and to query data. However, there are two disadvantages to consider, the translation of the query to the Java language to create the MapReduce job increases response time and can't perform all the queries with SQL. In addition, not support indexes or transactions.

AWS

Cloud computing or simply *cloud* provides computing resources and applications on demand via internet communications and whose total price of use is done according to consumption [42].

Cloud is able to offer low cost fast and flexible access to a variety of resources without investing in the acquisition of equipment or having to worry about their management. For this reason, the facilities offered by cloud computing is based on immediate access to resources and pay only for the use them [42].

Amazon Web Services (AWS), is responsible for maintaining equipment while the user only provisioned resources through a web application. The set of services offered are based on services computing, storage, databases, analysis, applications and

implementations that facilitate the progress of development, reduce costs and scalability of applications [42].

AWS offers numerous services and products (see Figure C), some of which are shown below:

- S3: Amazon Simple Storage Service (Amazon S3), offers storage capacity, durable and highly scalable. It is characterized by ease of use both when storing as to retrieve data from anywhere on the web. It also provides various types of settings regarding the time of storage in terms of access to data [43].
- EC2: Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity depending on user needs. The user has full control of its resources and it can be run on different environments Amazon. Because of this, the turnaround time is reduced and the starting time too [44].
- EMR: Amazon Elastic MapReduce (Amazon EMR), is related to faster processing of large amounts of data through a managed Hadoop framework. This provides the ability to distribute and process data between instances EC2 that can be scalable quickly and easily [45].

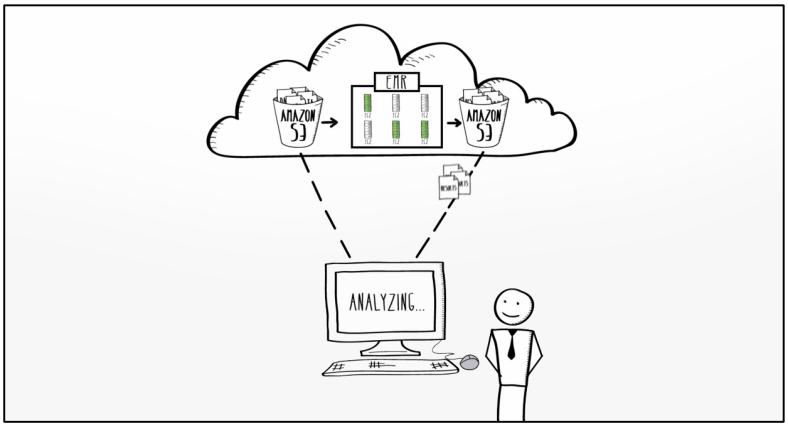


Figure C. Products: S3, EC2 y EMR. (Figure taken from [45])

System design

Firstly, it should be noted that the big data system has only one type of user, the system administrator, but it could restrict access to any of the system's functions depending on the type of user. In this project, it hasn't been considered relevant the distinction of roles and it depends on who exploit the system and how to exploit it, that is, there aren´t definite and fixed types of users to act on each of the parts of the system.

In our design, taking into account the above about the types of user, the system administrator has the ability to perform the loading, processing and visualization of data. The decomposition of the functions that characterize the system give rise to the three core operating system (see Figure D):

- **Load data**: The user is able to load the system data files with the traces to be processed. This project traces relate to travel of taxis in New York. Usually, this data can come from different sources, although it is not the case, so it cannot guarantee the initial homogeneity in both data format and structure.
- **Data processing**: Data must undergo a process of transformation and cleaning. This core is to format the data and discard those data that are wrong because of business rules and manipulations required by the target system. Records from the taxis have no direct transformations on the fields of the trace, but new fields are created from existing ones and considered invalid records are deleted.
- **Data Visualization**: The representation of the data after being processed can follow two way of interpretation. You can have a representation of results of processing, in other words, it could find the result before rendering or you can proceed to the exploitation of data to obtain non-trivial knowledge what is called *datamining*. One goal of the project is the visualization of data after processing, in particular, want to see the ten most frequent routes for hours each day, half days each day and each day. For this reason, we understand that we are in the first way of representation (pre-displayed results).

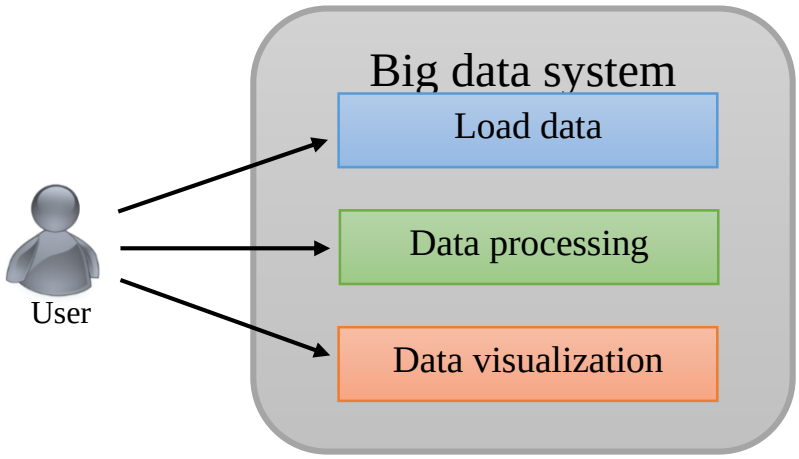


Figure D. General outline of the functions of a big data system.

The system will require a number of components that must interact with each other in order to continue their activity stream (see Figure E). The set of components are following:

- **Data Extraction**: It is the module responsible for obtaining the data from the various source systems. The data format is not standardized as a result of the data come from different sources, so that the data have to be processed. The source of data for this project is the DEBS 2015 Grand Challenge [2] and it is a single file with traces of travel for New York taxis.
- **Data Entry**: Collect the user's request what data must be loaded into the *data store*. Just load a file with all the data for travel.

- **Data store**: This is the component responsible for keeping both the data to be processed, related to trips made by taxis in New York, as the data once they have been reformatted and cleaned. This component interacts with the *cluster* to load the original data, reformat, delete those that are not valid and store data after being processed. Another line of connection is to the *delivering results* module to get the results and represent them later.
- **Cluster**: Performs tasks and cleaning transformation. Connect to the *data store* to export the processed data and the *user*, which will be detailed later.
- **Delivering results**: This component mediates between the *data store* and the *user*, so that the user can have the data ready to be painted.
- **Painted data**: Its function is the representation of results that are the most common routes in different time intervals. However, you might have data without seeking a specific result and interpret using this display, in other words, we could not know that there are frequent routes and once painted all points would see subsequently that there are indeed areas of greatest activity. Finally, the user enters the data for display.
- **User**: It remembers that the user only has the role of administrator, and to distinguish various types of user will be validated the type of user before interacting with any of the system components. The user connected to *data extraction* to obtain the original data. It also interacts with the *data store* for storing data from the extraction and the *cluster* to start the process of transformation and elimination. Once the data have been processed, the user receives it through the delivering results module. And finally, the user enters the data results in *painted data* module and display it.

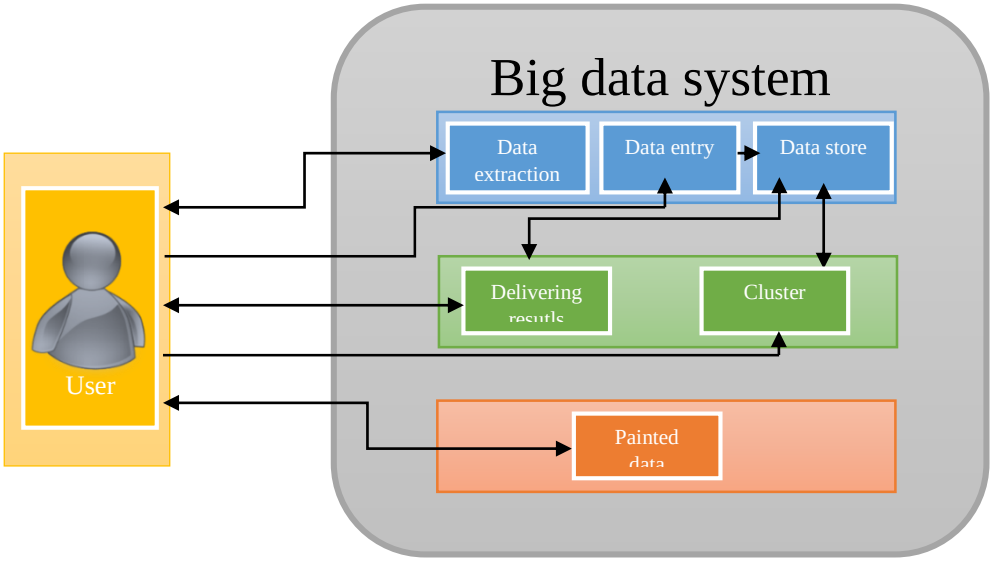


Figure E. Big data system, components and connections.

Testing and results

Environment test

After designing and implementing big data system, we proceed to test it in different environments in order to identify and meet the scalability limitations of the system. These tests are based on the size of the file to be inserted into the *data store* and the various configurations of the environment.

Both tests are complementary, that is, they are not performed separately. For each file size will be tested in different configurations VirtualBox environment (the first environment). For the AWS environment (the second environment) will be different, as will be detailed later.

The goal is to determine what should be the size of the file that would have to process each node in the cluster and as a result the time it would take to it. For the carrying out, it will double the size of the file is calculated based on the number of traces that contains, not its size in bytes. The following table shows the test files used, its number of traces and its size.

Name	Number of traces	Size(Kbytes)
0_data.csv	0	0
1_data.csv	1	1
50_data.csv	50	10
100_data.csv	100	19
200_data.csv	200	38
400_data.csv	400	75
800_data.csv	800	150
1600_data.csv	1600	300
3200_data.csv	3200	600
6400_data.csv	6400	1.200
12800_data.csv	12800	2.402
25600_data.csv	25600	4.809
51200_data.csv	51200	9.620
102400_data.csv	102400	19.232
204800_data.csv	204800	38.437
409600_data.csv	4096	76.867
819200_data.csv	819200	153.694
1638400_data.csv	1638400	307.331
3276800_data.csv	3276800	614.607
6553600_data.csv	6553600	1.229.357

Table 1. Test files.

As seen in Table 1, the first three files are not twice the previous one because it has been assumed that files are too small to vary their behavior.

Regarding testing VirtualBox, there have been two different configurations of the virtual machine. The two configurations are virtually identical, the only difference between them is the base memory. One of the configurations has 1GB of RAM and the other has 1.7GB of RAM.

In AWS, for cost reasons in the use of services they have been reduced testing in the *cloud*. It has chosen a test file and four clusters, which are formed by: 1 node, 2 nodes, 4 nodes and 8 nodes.

Results

In Figure F, the times and speeds for each file in the virtual machine with1GB of RAM is showed. We do not have results for a file twice the last because the process fails to complete. It is noted that the 204,800 records processed until the time is almost the same. This means that the system is unprofitable to this file size, as the cost in time to process 0 traces or 204,800 traces is almost equal. However, when we process twice records (405,600 records), there is abnormal behaviour, as it lowers the curve of time but from then time begins to increase doubled for the following files. For this reason, from 405,600 records file size and runtime grow almost in the same proportion.

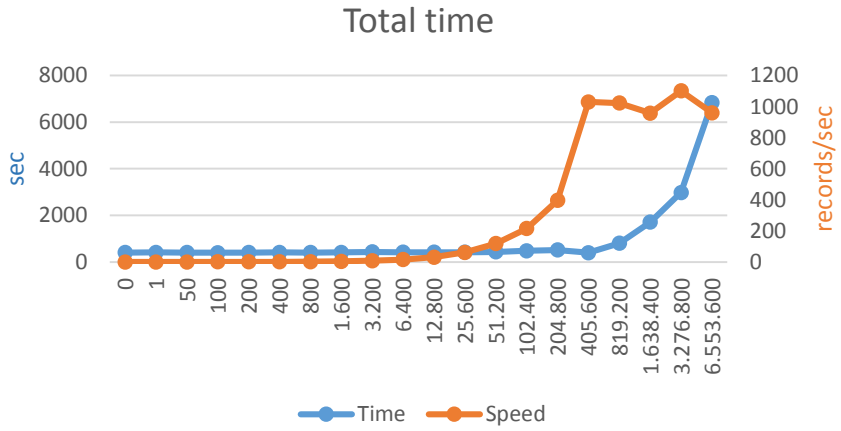


Figure F. Time and speed of first VirtualBox configuration for total process.

Speed has the same behavior, it increased to 204,800 records and it appears to be stable when the running time is almost double for each file.

For the second configuration of the virtual machine in its corresponding graph (see Figure G) it can be seen that the rate seems to be shifted to the right with respect to that of Figure F. The timeline remains stable until traces 204,800. It presents the same

abnormal behaviour previously discussed in registers 405,600, there is a drop from which growth begins. Unlike the graph of Figure F, a twice increase file size and the execution time is not proportional since it is observed that the times are not about twice the previous. This is because the increased RAM makes it can process as many records in less time. In relation to speed, in this configuration there is no stability.

For the same reason as before, there is no data for larger files than the last.

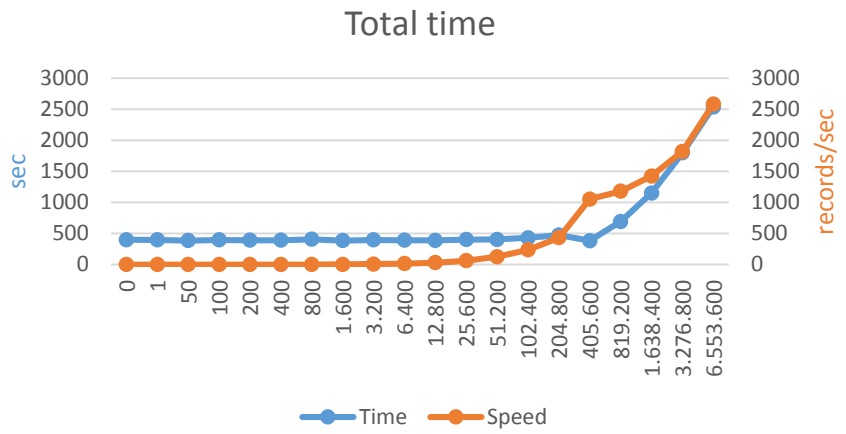


Figure G. Time and speed of second VirtualBox configuration for total process.

At the conclusion of the two graphs, we can say that the RAM of the environment directly affects execution times, but time and RAM is not proportional. Also, another conclusion is that for the test environment VirtualBox maximum file which can be processed is between 6,553,600 and 13,107,200 records it had been impossible to process the latter.

Machines AWS environment are 7 GB of RAM, so the timeline will grow so slowly that it will be practically constant. For this reason the execution times for the selected file is similar to the runtime of files with 0 records and 405,600 records in the Figure f and Figure g.

Regarding the relationship between time and the number of nodes (see **¡Error! No se encuentra el origen de la referencia.**) when nodes 1 and 2 runtime increases by sending data between cluster nodes. However, as we increase the nodes to 4 or 8 time decreases. This is because the computing power that is added to the system time offset cost of sending data between nodes in the cluster.

The conclusion is that to determine the number of nodes that must have the cluster system has to find a balance between latency introduced by the communication between nodes and computing capacity added.

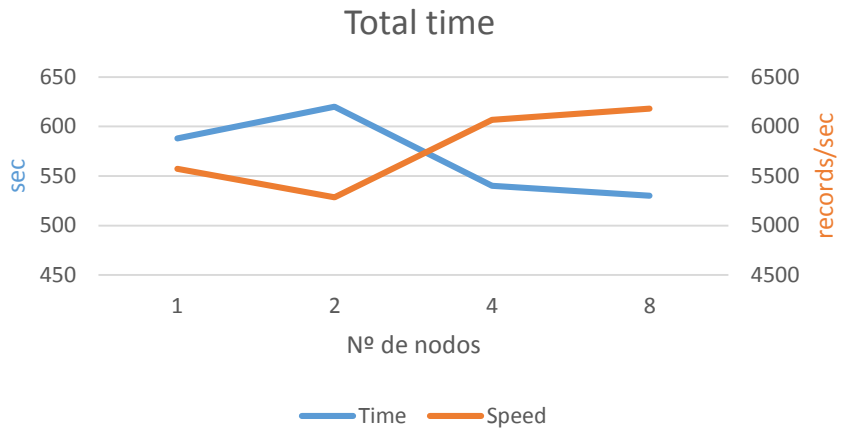


Figure H. Time and speed in AWS environment for total process.

Conclusions

The purpose of this Final Project was to develop a big system data to offer the user the possibility to interact with it. The user can load, process and display the data. In this case, the traces relate to travel of taxis in New York, and the end result is the representation of the most common routes for hours each day, half days each day and each day.

After making the design and implementation of the system it can be concluded compliance with specified requirements. The user has the ability to load data into HDFS with the ease of use that provides communication via the command console. Also the user performs data processing away from the complexity involved in MapReduce programming through the use of technology Hive. Finally, we note the main advantage of using QGIS to display the data that is to be a tool open source and freely available.

Future lines work

Some ideas for future work to expand system functionality or more testing it are:

- Viewing statistics from processing data. In this case, being taxi routes these statistics may be the percentage of customers using each method of payment (cash or card), the use of taxis by the hour or the average distances traveled throughout the day.
- Extraction behavior patterns depending on external parameters, in other words, combining data from taxi routes with climate data or data from social events.
- Using traces with different nature, not related to traffic.

Anexo D: Normativa y marco regulador

En relación al uso de las nuevas tecnologías, no existen inconvenientes legales directos. Sin embargo, pueden existir problemas de violación de la privacidad de las personas. De tal forma que estos problemas son tratados por los organismos correspondientes.

Toda aquella información que posibilite la identificación directa o indirectamente de cualquier persona se considera un dato de carácter personal. Por esto, el derecho fundamental a la protección de datos consiste en otorgar al ciudadano la capacidad de disponer, controlar y decidir sobre sus datos personales [48].

A nivel nacional es la Agencia Española de Protección de datos es la autoridad de control independiente encargada de velar por el cumplimiento de la normativa que hace referencia a la protección de datos. Además, asegura y protege el derecho fundamental a la protección de datos personales [48].

La Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de carácter personal tiene como objeto garantizar y proteger todo lo relacionado con el tratado de los datos personales , las libertades públicas y los derechos fundamentales de las personas físicas así como el honor e intimidad personal y familiar [49]. Y los derechos que se engloban en esta ley son [50]:

- Derecho de información: Cuando se procede a la recogida de datos el interesado tiene que ser informado.
- Derecho de acceso: El interesado puede conocer y obtener de forma gratuita los datos de carácter personal que van a ser tratados.
- Derecho de rectificación: Se permite la corrección de errores o la modificación de datos que sean inexactos o incompletos.
- Derecho de cancelación: Se puede suprimir los datos considerados inadecuados o excesivos.
- Derecho de oposición: Derecho del afectado a que no puedan ser tratados sus datos personales.

En el ámbito europeo, se han fortalecido los derechos de los ciudadanos adaptando las reglas para los negocios a consecuencia de la en la era digital en la que nos encontramos. Se ha considerado que las empresas no pueden compartir datos de los usuarios sin una previa autorización en la que ofrecen su consentimiento al intercambio de datos. Y en el caso de violación de los derechos de privacidad de sus usuarios la multa a la que tendrán que enfrentarse las empresas puede ascender al 4% de los ingresos de la compañía [51].

Referencias

[1] KAMBATLA, Karthik, et al. Trends in big data analytics. Journal of Parallel and Distributed Computing, 2014, vol. 74, no 7, p. 2561-2573.

[2] The 9th ACM International Conference on Distributed Event-Based Systems. URL:

http://www.debs2015.org/call-grand-challenge.html [13 de Febrero del 2016]

[3] Página oficial de Apache Hadoop. URL:

http://hadoop.apache.org/ [13 de Febrero del 2016]

[4] Página oficial de Apache Hive. URL:

https://hive.apache.org/ [13 de Febrero del 2016]

[5] Página oficial de IBM. URL:

http://www.ibm.com/es-es/ [13 de Febrero del 2016]

[6] IBM. Bringing big data to the enterprise. URL:

http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html [13 de Febrero del 2016]

[7] Miguel Ángel Lucas. Big Data, significado y su utilidad en la sociedad. URL:

http://mibloguel.com/big-data-significado-y-su-utilidad-en-la-sociedad/ [13 de Febrero del 2016]

[8] El economista. La moda del Big Data: ¿En qué consiste en realidad?. URL:

http://www.eleconomista.es/tecnologia/noticias/5578707/02/14/La-moda-del-Big-Data-En-que-consiste-en-realidad.html [13 de Febrero del 2016]

[9] O'Reilly Media, Inc. Big Data Now: 2012. " O'Reilly Media, Inc.", 2012.

[10] Juan Vidal. Big data: Gestión de datos no estructurados. URL:

http://www.dataprix.com/blog-it/big-data/big-data-gestion-datos-no-estructurados [13 de Febrero del 2016]

[11] IBM. Analytics: el uso de big data en el mundo real. URL:

http://www-05.ibm.com/services/es/gbs/consulting/pdf/El_uso_de_Big_Data_en_el_mundo_real.pdf [13 de Febrero del 2016]

[12] ABC. Big data: ¿vidas privadas al alcance de todos?. URL:

http://www.abc.es/tecnologia/informatica-software/20131028/abci-entrevista-data-201310221252.html [13 de Febrero del 2016]

[13] Tendencias 21. ¿Qué es la comunicación M2M?. URL:

http://www.tendencias21.net/telefonica/Que-es-la-comunicacion-M2M_a801.html [13 de Febrero del 2016]

[14] ZIKOPOULOS, Paul, et al. Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.

[15] Página oficial Inditex. URL:

http://www.inditex.com/es/home [14 de Febrero del 2016]

[16] El secreto de Zara: la tecnología. URL:

http://www.emprendemania.com/2015/06/el-secreto-de-zara-la-tecnologia.html [14 de Febrero del 2016]

[17] Página oficial de BBVA. URL:

https://www.bbva.es/particulares/index.jsp [14 de Febrero del 2016]

[18] BBVA Inovation Center. Proyecto Big Data. URL:

http://www.centrodeinnovacionbbva.com/proyectos/big-data [14 de Febrero del 2016]

[19] El mundo. Predecir en un 70% un crimen futuro utilizando 'big data'. URL:

http://www.elmundo.es/economia/2015/01/21/54be9d42ca474188098b456f.html [14 de Febrero del 2016]

[20] CESAR, Pérez López; DANIEL, Santín González. Minería de Datos. Técnicas y Herramientas. Editorial: Thomson, 2007.

[21] Microsoft. Conceptos de minería de datos. URL:

https://msdn.microsoft.com/es-es/library/ms174949(v=sql.120).aspx [14 de Febrero del 2016]

[22] Sinnexus. Datamining (Minería de datos). URL:

http://www.sinnexus.com/business_intelligence/datamining.aspx [14 de Febrero del 2016]

[23] César Krall. Minería de datos: ¿qué es? ¿para qué sirve?. URL:

http://www.aprenderaprogramar.com/index.php?option=com_content&id=252:mineria-de-datos-data-mining-ique-es-ipara-que-sirve-1o-parte-dv00105a&Itemid=164 [14 de Febrero del 2016]

[24] Fases de minería de datos. URL:

http://exa.unne.edu.ar/depar/areas/informatica/dad/DAD/Presentaciones/Mineria_de_Datos.pdf

[25] COULOURIS, George F.; DOLLIMORE, Jean; KINDBERG, Tim. Distributed systems: concepts and design. pearson education, 2005.

[26] EcuRed. Computación distribuida. URL:

http://www.ecured.cu/Computaci%C3%B3n_distribuida [14 de Febrero del 2016]

[27] Juan Pablo Bustos Thames. Arquitectura de sistemas distribuidos. URL:

http://es.slideshare.net/jpbthames/arquitectura-de-sistemas-distribuidos [14 de Febrero del 2016]

[28] Javier Condori Flores. Cluster Beowulf. URL:

http://es.slideshare.net/JacF/cluster-beowulf-javier-condori-flores?qid=425e16f6-9e87-436a-b2f0-fbd75d49e882&v=&b=&from_search=7 [14 de Febrero del 2016]

[29] Academica. Clúster Informático: Un resultado no planeado de las Matemáticas. URL:

http://www.academica.mx/blogs/cl%C3%BAster-inform%C3%A1tico-un-resultado-no-planeado-las-matem%C3%A1ticas [14 de Febrero del 2016]

[30] El diario de Ferdy. El teorema de CAP. URL:

http://www.rodenas.org/ferdyblog/2011/02/25/el-teorema-de-cap/ [14 de Febrero del 2016]

[31] Enrique Amodeo. noSQL (2): No necesitas ACID. URL:

https://eamodeorubio.wordpress.com/2010/05/17/nosql-2-no-necesitas-acid/ [14 de Febrero del 2016]

[32] DEAN, Jeffrey; GHEMAWAT, Sanjay. MapReduce: simplified data processing on large clusters. Communications of the ACM, 2008, vol. 51, no 1, p. 107-113.

[33] Tomás Fernández Pena. Big Data y Mapreduce. URL:

http://es.slideshare.net/tfpena/map-reduce-csd?qid=62a2eec4-c9f3-42e3-934c-1fe6f26a597e&v=&b=&from_search=2 [14 de Febrero del 2016]

[34] Tutorials Point. Hadoop – Mapreduce. URL:

http://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm [14 de Febrero del 2016]

[35] WHITE, Tom. Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.

[36] Tutorials Point. Hadoop – Introduction. URL:

http://www.tutorialspoint.com/hadoop/hadoop_introduction.htm [14 de Febrero del 2016]

[37] Formación Hadoop. Curso: Desarrollador para Apache Hadoop. URL:

http://formacionhadoop.com/aulavirtual/pluginfile.php/32/course/summary/Desarrollador%20Apache%20Hadoop%20Cap%C3%ADtulo%203%20-%20Hadoop%20Conceptos%20B%C3%A1sicos.pdf [14 de Febrero del 2016]

[38] Tutorials Point. Hadoop – HDFS overview. URL:

http://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm [14 de Febrero del 2016]

[39] Tomás Fernández Pena. Apache Hadoop. URL:

http://es.slideshare.net/tfpena/hadoop-notas?qid=a0b0f127-ec8b-4e43-ad70-1bac43045d60&v=&b=&from_search=1 [14 de Febrero del 2016]

[40] CAPRIOLO, Edward; WAMPLER, Dean; RUTHERGLEN, Jason. Programming hive. " O'Reilly Media, Inc.", 2012.

[41] Juan Alonso Ramos. Introducción a Apache Hive, el datawarehouse de Hadoop. URL:

http://www.adictosaltrabajo.com/tutoriales/hive-first-steps/ [14 de Febrero del 2015]

[42] Amazon Web Services. ¿Qué es la informática en la nube?. URL:

https://aws.amazon.com/es/what-is-cloud-computing/?sc_channel=PS&sc_campaign=acquisition_ES&sc_publisher=google&sc_medium=cloud_computing_b&sc_content=sitelink&sc_detail=amazon%20web%20service&sc_category=cloud_computing&sc_segment=why_amazon_web_services&sc_matchtype=p&sc_country=ES&s_kwcid=AL!4422!3!91479926601!p!!g!!amazon%20web%20service&ef_id=VruyqQAABfvUwIsh:20160210215905:s [14 de Febrero del 2016]

[43] Amazon Web Services. Amazon S3. URL:

https://aws.amazon.com/es/s3/ [14 de Febrero del 2016]

[44] Amazon Web Services. Amazon EC2. URL:

https://aws.amazon.com/es/ec2/ [14 de Febrero del 2016]

[45] Amazon Web Services. Amazon EMR. URL:

https://aws.amazon.com/es/elasticmapreduce/ [14 de Febrero del 2016]

[46] Mapping GIS. ¿Por qué QGIS es la referencia en los clientes SIG open source?. URL:

http://mappinggis.com/2014/06/por-que-qgis-es-la-referencia-sig-open-source/ [14 de Febrero del 2016]

[47] Mercedes Alcívar, Iván Espinoza, Vanessa Cedeño. Análisis de la información de una base de Datos Transaccional usando Hive sobre Hadoop. URL:

https://www.dspace.espol.edu.ec/bitstream/123456789/20816/1/Base%20de%20Datos%20Transaccional%20Usando%20Hive%20sobre%20Hadoop.pdf [14 de Febrero del 2016]

[48] Agencia española de protección de datos. Tu derecho fundamental a la protección de datos. URL:

https://www.agpd.es/portalwebAGPD/CanalDelCiudadano/derechos/index-ides-idphp.php [13 de Febrero del 2016]

[49] Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal. URL:

http://noticias.juridicas.com/base_datos/Admin/lo15-1999.t1.html#t1 [13 de Febrero del 2016]

[50] Agencia española de protección de datos. Principales derechos. URL:

https://www.agpd.es/portalwebAGPD/CanalDelCiudadano/derechos/principales_derechos/index-ides-idphp.php [13 de Febrero del 2016]

[51] El país. La UE aprueba la ley de protección de datos, bloqueada desde 2013. URL:

http://internacional.elpais.com/internacional/2015/12/15/actualidad/1450208377_400556.html [13 de Febrero del 2016]

[52] BASANTA-VAL, P., et al. Improving the predictability of distributed stream processors. Future Generation Computer Systems, 2015, vol. 52, p. 22-36.