Mapreduce para archivos binarios distribuidos

Cynthia Alejandra Martínez Pinto
Instituto Tecnológico de Ciudad Guzmán
cynthia_amp@hotmail.com

Rosa María Michel Nava
Instituto Tecnológico de Ciudad Guzmán
michel91_3@hotmail.com

Ruth Clementina Barragán López
Instituto Tecnológico de Ciudad Guzmán
ch2nita_mex@hotmail.com

Resumen

El flujo de datos generados diariamente se ha incrementado exponencialmente, al grado que resulta importante aprender a manejar y manipular esta información para no quedar rezagado. Grandes empresas como Oracle, IBM y Microsoft (por mencionar algunas), le están apostando al negocio del almacenamiento y análisis de esta abundante información.

La tendencia es hacer más accesible la manipulación de grandes datos (Big Data), donde Hadoop se ha convertido en un estándar para el manejo de esta información. Ha resultado muy efectivo para analizar datos No Estructurados en la nube, pero esto apenas comienza y en el campo de la astrofísica esta herramienta no es suficiente.

La técnica de MapReduce, es muy efectiva para el análisis de archivos de texto, sin embargo cuando se utiliza con archivos binarios distribuidos los resultados que se arrojan son erróneos. Por tal motivo, doctores de la Universidad de California, Riverside; trabajan en un framework denominado StratOS, que soluciona entre otras cosas, el manejo de este tipo de archivos.

Como colaboradoras de esta investigación, el presente artículo trata del desarrollo de un software basado en la técnica de MapReduce, que analiza archivos binarios distribuidos, presentando el resultado obtenido a través de una interfaz desarrollada en Python.

Palabras Claves: MapReduce, StratOS, Big Data.

Introducción

La gran cantidad de datos que cada día se generan, capturan, almacenan y analizan, han dado lugar a una nueva manera de ver y manipular la información. Los grandes datos o macrodatos (Big Data) son una realidad consolidada. En un estudio que realizó la empresa IDC en el 2012 se informó que se crearon 2,8 zettabytes(ZB) de datos sólo en ese año y se calcula que esta cifra se dobla cada dos. Así mismo el estudio revela que menos del 1% de los datos del mundo se analizan y el 20% de los datos no están protegidos. (Joyanes Aguilar, 2013)

Los problemas que el Big Data ofrece, son un gran reto para arquitecturas tradicionales, basadas en sistemas de cómputo desacoplados del sistema de almacenamiento. Esto provoca cuellos de botella al intentar analizar información en forma simultánea, principalmente por las Entradas/Salidas de datos y no por el CPU.

Una alternativa para resolver este problema es el método desarrollado por Google a través de la plataforma distribuida denominada Hadoop, la cual incluye el paradigma de MapReduce donde, problemas masivos son separados en partes pequeñas y fáciles de analizar (Map) para después agregar los resultados parciales en el resultado final (Reduce). Esta arquitectura consta de diferentes nodos, cada uno con un CPU y gran cantidad de espacio de almacenamiento local, conectados por una red de velocidad media. Esto reduce los requerimientos de ancho de banda de la red y el tiempo de acceso a los datos, eliminando el cuello de botella de arquitecturas tradicionales.

A pesar de la gran aceptación de Hadoop para análisis de grandes volúmenes de información su utilidad en ciencias es reducida. Esto debido a la pronunciada pendiente en la curva de aprendizaje de Hadoop,

el uso casi exclusivo de java (considerado de bajo rendimiento para aplicaciones científicas) y un sistema de archivos muy limitado (*append-only* y *block size* fijo).

En estudios científicos realizados por astrofísicos de la Universidad de California, Riverside, se tienen problemas con datos binarios distribuidos, los cuales deben ser analizados usando programas específicos para astronomía. A pesar de esto, no existen patrones como MapReduce implementados para ese tipo de aplicaciones.

El objetivo de la presente investigación fue desarrollar una interfaz en Python para ejecutar un programa que analice datos binarios distribuidos en un clúster usando el sistema StratOS. Un ejemplo específico de la utilización del software desarrollado es el acceso a partículas en una simulación en diferentes tiempos. La interfaz de Python ejecuta un programa en C en varios archivos y recolecta los datos de salida de cada archivo ordenándolos antes de presentarlos al usuario como un arreglo en Python.

CONTENIDO

FUNDAMENTOS TEÓRICOS

Para la realización de este proyecto, se ha investigado y profundizado en los siguientes conceptos.

Big Data

Es “el término que se aplica a conjuntos de datos cuyo volumen supera la capacidad de las herramientas informáticas de uso común, para capturar, gestionar y procesar datos en un lapso de tiempo razonable. Los volúmenes de Big Data oscilan entre decenas de terabytes y petabytes”. (Joyanes Aguilar, 2013)

Algunas características son:

- Volumen: el tamaño de los datos determina el valor y el potencial en cuestión.
- Variedad: significa a que categoría pertenecen pues ayuda a su análisis.
- Velocidad: es la velocidad de la generación de los datos o la rapidez con que estos se generan y procesan para satisfacer las demandas.
- Veracidad: la calidad de los datos en cuestión.

Hadoop

Apache Hadoop es una biblioteca de código abierto (open source) que soporta el procesamiento distribuido de grandes conjuntos de datos a través de miles de computadoras ordinarias.

Hadoop es líder en plataformas de Big Data y su uso crece de modo exponencial. Éste consta con tres componentes principales: Hadoop Distributed File System (HDFS), MapReduce y Hadoop Common. (Joyanes Aguilar, 2013)

La eficiencia de esta plataforma consiste en su arquitectura, la cual utiliza para reducir el tráfico de datos, una red troncal donde cada sistema de archivos conoce y proporciona su ubicación y donde está el nodo trabajador; mientras que el sistema de archivos HDFS, utiliza esta información en la replicación de datos, para conservar copias de los datos en racks diferentes.

Un clúster típico de Hadoop incluye:

- Nodo maestro: que cuenta con un rastreador de trabajo, un rastreador de tareas, un nodo de nombres y un nodo de datos.
- Nodo esclavo: que contiene un nodo de datos y un rastreador de tareas.

Hadoop Distributed File System (HDFS)

Es un Sistema distribuido, portátil y escalable escrito en Java para Hadoop. Cada nodo en una instancia Hadoop, tiene un único nodo de datos y un clúster de datos. Cada nodo sirve bloques de 64 MB sobre la red, usando un protocolo específico para HDFS. El sistema de archivos usa la capa TCP/IP para la comunicación; los clientes usan RPC (Llamada a Procesamiento Remoto) debido a que permite a un programa ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

MapReduce

Es el modelo de programación utilizado para dar soporte a la comunicación paralela sobre colecciones de datos en grupos de computadoras.

Es una metodología muy efectiva que agiliza el acceso a los datos dentro del clúster, sin embargo no se presta para todo tipo de información. Principalmente trabaja de la siguiente forma:

Funcion Map()

Esta función se encarga del mapeo y es aplicada en paralelo para cada ítem en la entrada de datos. Esto produce una lista de pares (k2,v2) por cada llamada. Después el framework de MapReduce junta todos los pares con la misma clave de todas las listas y los agrupa, creando un grupo por cada una de las diferentes claves generadas.

Funcion Reduce()

Esta función produce un valor v3 o una llamada vacía, aunque una llamada puede retornar más de un valor. El retorno de todas esas llamadas se recoge como la lista de resultado deseado. Por lo tanto el framework MapReduce transforma una lista de pares en una lista de valores.

Ingeniería del software

La ingeniería del software permite al diseñador de programas, realizar su tarea de construcción de software como un problema de ingeniería haciendo uso de guías, principios y normas que le permitirán el correcto desarrollo de su labor. Adicionalmente, dispondrá de un conjunto de herramientas que le permitirán la evaluación, validación, depuración y corrección del software desarrollado.

Etapas del ciclo de vida del software

El ciclo de vida clásico del software siendo uno de los más utilizados tal como lo plantean diferentes autores, consta de las siguientes etapas:

- **Análisis:** en esta etapa se debe entender y comprender de forma detallada cuál es la problemática a resolver, verificando el entorno en el cual se encuentra dicho problema, de tal manera que se obtenga la información necesaria y suficiente para afrontar su respectiva solución. Esta etapa es conocida como la del “qué se va a solucionar”.
- **Diseño:** una vez que se tiene la suficiente información del problema a solucionar, es importante determinar la estrategia que se va a utilizar para resolver el problema. Esta etapa es conocida bajo el “cómo se va a solucionar”.
- **Codificación:** partiendo del análisis y diseño de la solución, en esta etapa se procede a desarrollar el correspondiente programa que solucione el problema mediante el uso de unas herramientas como lo es Java, Php y MySQL.

- **Pruebas:** los errores humanos dentro de la programación de las computadoras son muchos y aumentan considerablemente con la complejidad del problema. Es necesario realizar las debidas pruebas que garanticen el correcto funcionamiento de dicho programa bajo el mayor número de situaciones posibles a las que se pueda enfrentar, esto es realizando diferentes tipos de pruebas como son las pruebas de caja blanca y pruebas de caja negra.
- **Mantenimiento:** una vez instalado un programa y puesto en marcha para realizar la solución del problema previamente planteado o satisfacer una determinada necesidad, es importante mantener una estructura de actualización, verificación y validación que permitan a dicho programa ser útil y mantenerse actualizado según las necesidades o requerimientos planteados durante su vida útil. Para realizar un adecuado mantenimiento, es necesario contar con una buena documentación del mismo.

StratOS

Es una plataforma que permite que un centro de datos pueda ser tratado como una sola computadora, esto hace que el proceso de escribir un programa en paralelo para un centro de datos no sea más complicado que escribir un script de Python para una computadora de escritorio.

Los usuarios pueden ejecutar aplicaciones pre-existentes de análisis de datos distribuidos en miles de computadoras con solo pulsar unas cuantas teclas. Software desarrollado por los doctores Nathaniel R. Stickley y Miguel Angel Aragón Calvo en la Universidad de California, Riverside.

METODOLOGÍA

Para realizar el presente proyecto se siguió una metodología basada en las etapas de Análisis y planeación, Diseño, Codificación y Pruebas. A continuación se describen las actividades realizadas dentro de cada una de estas etapas.

1. Análisis

Dentro de la primera etapa de análisis se investigó sobre la plataforma de desarrollo, las características del clúster donde se instaló el software y el paradigma de la programación en paralelo.

Así mismo se hizo un análisis de características y comportamientos del sistema, y los riesgos que pueden amenazar el proyecto, en la tabla 1 se listan algunos de los riesgos analizados.

Riesgo	Categoría del riesgo	Descripción
Enfermedad de la persona	Proyecto	La persona abandona el proyecto antes de que finalice.
Cambio de requerimientos	Proyecto y producto	Habrá más cambios en los requerimientos de lo esperado.
Subestimación del tamaño	Proyecto y producto	El tamaño del sistema se ha subestimado.
Tecnología desconocida	Proyecto y producto	Se desconoce la tecnología requerida para realizar el proyecto

Tabla 1 Riesgos del proyecto

Al presentarse estos riesgos durante la elaboración del proyecto, se tomaron las medidas más adecuadas para poder terminar con éxito los objetivos planteados.

2. Diseño

En este proceso se traducen los requisitos en una representación del software de forma que pueda conocerse la arquitectura, funcionalidad y calidad del mismo antes de comenzar la codificación. El diseño abarcó la estructura de los datos, arquitectura de las aplicaciones, estructura interna de los programas e interfaces. A continuación se describe lo más relevante.

2.1 Diagrama de componentes

Los diagramas de componentes sirven para mostrar los elementos de diseño de un sistema de software, ya que permite visualizar con mayor facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan.

En la figura 1 se muestra el diagrama de componentes del sistema "Software para analizar imágenes astronómicas en forma paralela y automática" que forma parte de esta investigación.

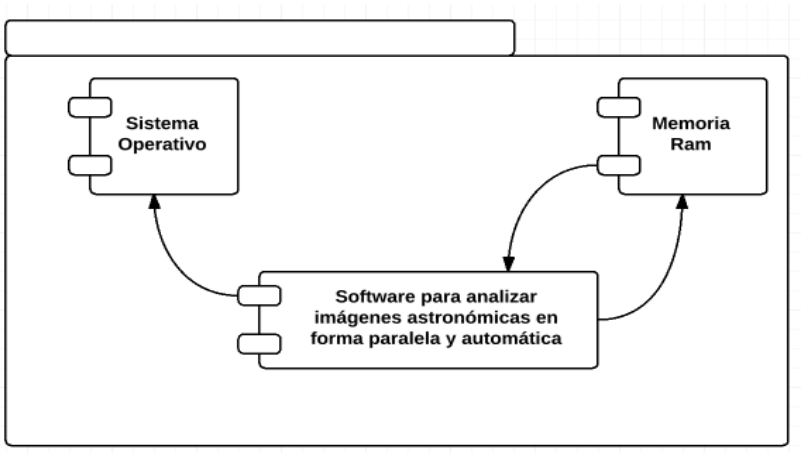

Figura 1. Diagrama de componentes del software para analizar imágenes astronómicas en forma paralela y automática

En la figura 1 se muestra que el software realizado necesita de la memoria Ram para ejecutar todas las operaciones, debido a que el clúster cuenta con 256 GB de memoria, se pidió que todos los resultados se desplieguen en memoria para una mayor velocidad en la presentación de resultados. El sistema operativo Linux es indispensable como soporte a la aplicación y Hadoop.

2.2 Arquitectura

El software para analizar imágenes astronómicas en forma paralela y automática trabaja bajo la plataforma Hadoop, donde ésta analiza la información y se la envía ya procesada al software; ambos programas se encuentran alojados dentro del sistema Operativo Linux, distribución Ubuntu, ya que es un software libre que permite realizar cambios idóneamente para administrar el clúster. Se accesa al cúster a través de Internet.

2.3 Interfaz

La interfaz permite al usuario comunicarse con el clúster de manera que éste pueda manipular los archivos de manera rápida y correcta, obteniendo de esta forma el análisis de la información. Para ello se decidió realizar una interfaz sencilla en la cual el usuario sólo tenga que ingresar los archivos para después correr la aplicación en el botón “Run”, de esta manera se realiza una validación rápida ya que sólo se necesita dar clic en botones y esperar la información en el recuadro tal como aparece en la figura 2.

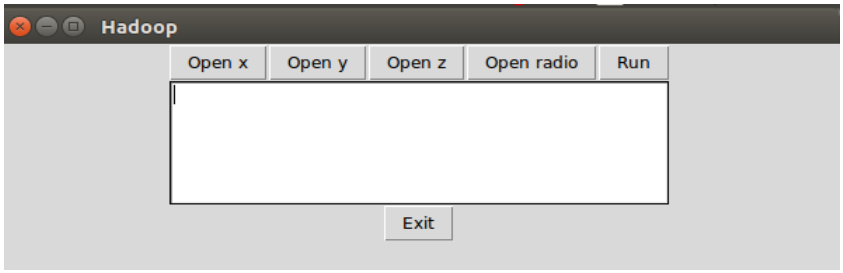

Figura 2. Interfaz de usuario del software para analizar imágenes astronómicas en forma paralela y automática

3. Codificación

El software resultante fue codificado en Java, otra parte en Python y una más en C++.

Para comprender mejor el funcionamiento de algunas funciones de Hapood, StratOS y MapReduce se desarrolló un programa para contar la concurrencia de las palabras en un archivo de texto con extensión ".txt".

El programa fué realizado utilizando el Modelo Vista Controlador por lo que se encuentra divido en tres clases, las cuales fueron desarrolladas en el lenguaje de Java debido a que este lenguaje es sencillo y mejor conocido.

Para poder utilizar MapReduce se usaron las siguientes funciones:

```
// Configuración de las clases Mapper y Reducer

    conf.setMapperClass(WordCountMapper.class);

    conf.setReducerClass(WordCountReducer.class);

// Se necesitarán dos argumentos uno para la ruta del documento de entrada y otra para la salida por lo que se declaran a continuación

    Path inp = new Path(args[0]);

    Path out = new Path(args[1]);
```

Y la función que los manda llamar es:

```
int res = ToolRunner.run(new Configuration(), new WordCount(),args);
```

4. Pruebas

Se realizaron las pruebas de rigor de caja blanca y caja negra, así mismo se probó el software de manera local y también en el clúster. El resultado final, todavía se encuentra en etapa de pruebas.

RESULTADOS

A continuación se describen las actividades que se realizaron para poder desarrollar el software con sus respectivos resultados.

a. Servidor local

Antes de poder ingresar a Hadoop se necesita tener Ubuntu ya sea de manera local o en un emulador y después tener instalado Hadoop.

En la figura 3 se muestra su acceso desde del navegador Web, esto se logra una vez que se han configurado todos los archivos desde la terminal en el sistema operativo Ubuntu.

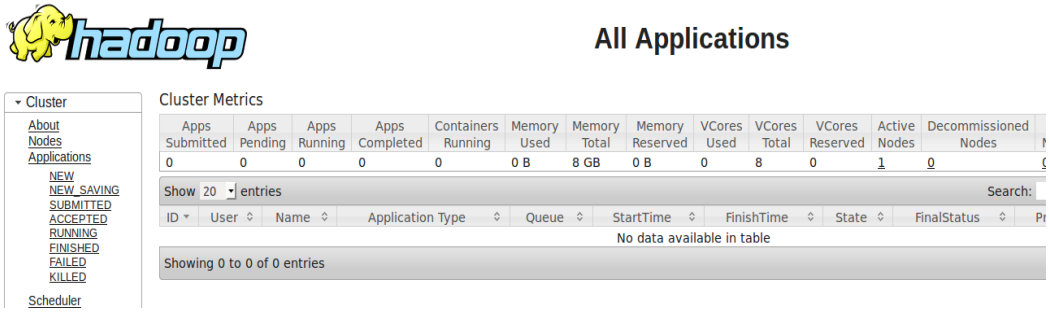

Figura 3. Acceso al clúster local a través del navegador

Se realizó un programa llamado WordCount que busca la concurrencia de las palabras en un archivo ".txt".

Este programa sirvió de apoyo para entender mejor el funcionamiento de Hadoop, StratOS y MapReduce; utilizando en el Modelo Vista Controlador, la herramienta Eclipse, que es un entorno de

desarrollo integrado (IDE) y el lenguaje de programación Java por ser el mejor conocido. En la figura 4, se puede apreciar la ejecución del contador del programa.

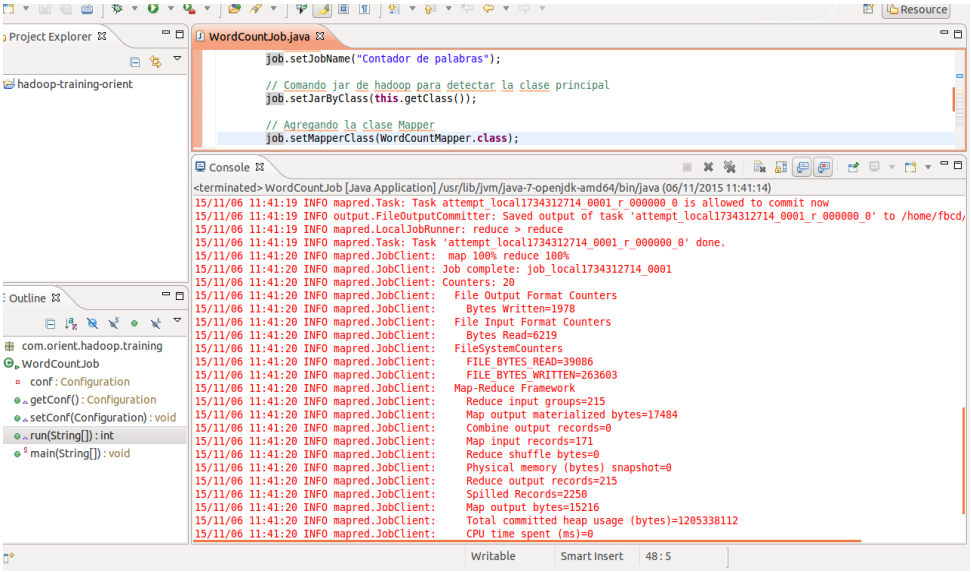

Figura 4. Ejecución del programa WordCount

Cuando el programa termina de ejecutarse arroja dos archivos de texto en el directorio donde se encuentra ubicado el ".jar":

- _SUCCESS.txt es un archivo vacío, que sirve para comprobar que no hubo errores durante la ejecución de la aplicación.
- part-r-00000.txt es un archivo que contiene una lista de las palabras analizadas acompañadas con números que indican la cantidad de veces que fueron encontradas en el archivo de entrada ingresado por el usuario.

En la figura 5 se puede apreciar el resultado que arroja el contador de palabras.

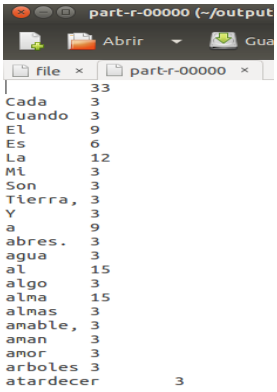

Figura 5. Conteo de la concurrencia de palabras

Aquí es necesario aclarar que es un Snapshots. Es una copia virtual de un conjunto de archivos, directorios o grandes volúmenes de datos. Las instantáneas o Snapshots se utilizan a menudo en sistemas de almacenamiento para mejorar la protección y eficiencia de los datos. En el proyecto se utilizan para obtener información de los archivos binarios leídos, de los cuales se obtiene la información a analizar.

Después de analizar el funcionamiento de Hadoop con el programa WordCount se elaboró un programa en Python para trabajar con los Snapshots, los cuales cuentan con cuatro valores x, y, z y radio. Una vez leídos y separados con estos valores se presentan los resultados en un arreglo de Python en la interfaz de la figura 2 (Sección 2.3 Interfaz).

b. Clúster de la Universidad

Después de haber comprobado que Hadoop funciona correctamente en forma local, se procede a ingresar al servidor apache de la Universidad mediante el comando SSH seguido del usuario como se muestra en la figura 6.

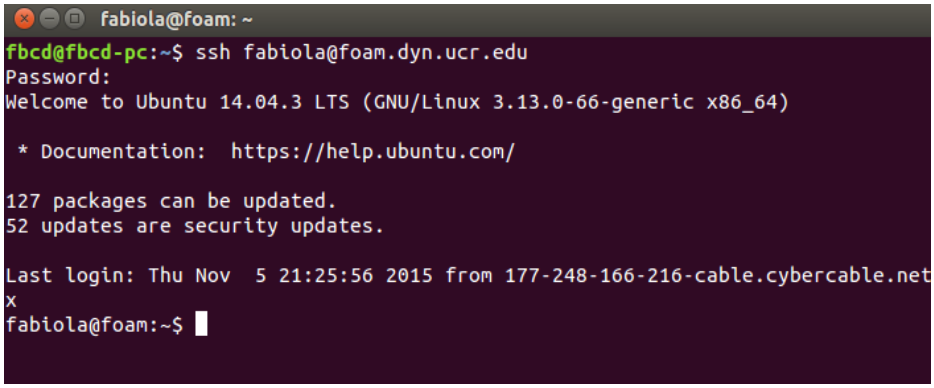

Figura 6. Ingreso al clúster con el comando ssh

Para ejecutar el programa realizado en Eclipse se subió el archivo .jar al servidor de la Universidad, esto se realiza a través de la terminal usando GitHub, el cual también se encuentra instalado en clúster, para poder almacenar archivos en el clúster primero se deben subir a GitHub con el comando Git Gui, si todo se encuentra instalado correctamente deberá aparecer una cuadro de diálogo llamado Git Gui.

Después de haber realizado el paso anterior se escoge alguna de las tres opciones que aparecen las cuales ayudarán a crear un nuevo repositorio, clonar o abrir uno existente. Se realiza un clon y se copia el software a uno de los discos del clúster.

Las siguientes etapas de pruebas y resultados del software alojado en la Universidad de California, Riverside no se muestran en el presente artículo, pues aún no ha sido liberado para su difusión.

CONCLUSIÓN

Los avances tecnológicos tienen como trasfondo horas de investigación y trabajo arduo. Este proyecto no es la excepción. Los resultados mostrados en el presente artículo son sólo avances de los pasos que se han seguido para llegar a la meta, que es obtener un software que presente de forma comprensible y ordenada los datos binarios distribuidos que han sido tomados de un clúster que utiliza como plataforma el sistema StratOS.

El trabajo colaborativo entre universidades es enriquecedor y es una gran experiencia el formar parte de la construcción de software que utiliza tecnología de última generación. Nuestro reconocimiento a los doctores de la Universidad de California, Riverside Aragón Calvo y Stickley por sus grandes aportaciones a la ciencia y les agradecemos el permitirnos ser colaboradoras de sus proyectos.

Como trabajo futuro queda el probar el software en diferentes clústeres, con archivos de distinto origen y confirmar que la obtención y despliegue de resultados es el mismo que en la Universidad de California, Riverside.

Bibliografía

Booch, Grady, Rumbaugh, James, Jacobson, Ivar. (2006) *El lenguaje unificado de modelado, segunda edición.* Madrid: Pearson Educación, S.A.

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

Pressman, R. S. (2010). *Ingeniería del software, un enfoque práctico.* Mc Graw Hill.

Sommerville, I. (2005). *Ingeniería del software.* Madrid: Pearson Educacion.

Anónimo 1. Hadoop. Consultado el 3 de Febrero de 2016 desde http://hadoop.apache.org/

Anónimo 2. Hadoop and Big Data. Consultado el 30 de Enero de 2016 desde http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html

Anónimo 3. MapReduce. Consultado el 5 de Febrero de 2015 desde https://www.google.com.mx/search?q=Mapreduce&oq=Mapreduce&aqs=chrome..69i57j0l5.3073j0j4&sourceid=chrome&es_sm=93&ie=UTF-8

IBM. What is MapReduce?. Consultado el 2 de Febrero de 2015 desde http://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/

EMC corporation. Consultado el 09 de Marzo de 2016. Estudio del universo digital de EMC con investigación y análisis de IDC. Desde http://mexico.emc.com/leadership/digital-universe/index.htm