ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"MÓDULO DE RECOMENDACIONES DE PÁGINAS A VISITAR EN LA WIKIPEDIA, BASADO EN LAS APORTACIONES EFECTUADAS POR LA COMUNIDAD DE USUARIOS USANDO HADOOP"

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

ESPECIALIZACIÓN: SISTEMAS TECNOLÓGICOS

Presentada por:

BOLÍVAR ALBERTO ELBERT PONTÓN

ANDRÉS MARTIN CANTOS RIVADENEIRA

Guayaquil - Ecuador

2009

AGRADECIMIENTOS

Agradezco a Dios por las bendiciones recibidas, a mis padres, por darme la estabilidad emocional, económica y sentimental; para poder llegar hasta este logro, que definitivamente no hubiese podido ser realidad sin ustedes, a todos mis amigos pasados y presentes; pasados por ayudarme a crecer y madurar como persona y presentes por estar siempre conmigo apoyándome en todo las circunstancias posibles, también son parte de esta alegría.

DEDICATORIA

A mí querida familia que siempre me ha apoyado y estimulado.

A mi tutor M. Sc. Cristina Abad y a los maestros de la ESPOL por los valiosos conocimientos adquiridos.

TRIBUNAL DE GRADO

M. Sc. Cristina Abad

DIRECTOR DE MATERIA

M. Sc. Marcelo Loor

PROFESOR DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la **Escuela Superior Politécnica del Litoral**"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Bolívar Alberto Elbert Pontón

Andrés Martin Cantos Rivadeneira

RESUMEN

Los *sistemas de recomendaciones* cada día se vuelven indispensables para filtrar la gran cantidad de información disponible en la Web. Uno de los objetivos principales de estos sistemas es asistir a los usuarios en sus procesos de búsqueda de información en la Web.

En este trabajo se presenta una revisión básica de los aspectos fundamentales relacionados con el diseño, implementación y estructura de un módulo de recomendación para la Wikipedia, utilizando *Map/Reduce* implementado en el *Framework* Apache Hadoop para procesar grandes cantidades de información.

En la Wikipedia se puede catalogar dos tipos de usuarios, el usuario aportador y el usuario que realiza la consulta, los usuarios suelen aportar a diversas *wikis* sean estas o no de la misma categoría o contenido.

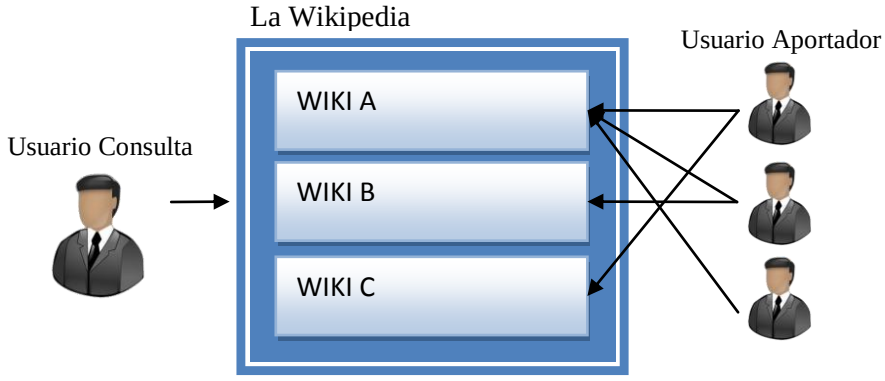

Figura 1. Rol de usuarios en la Wikipedia

Se creó el módulo de recomendaciones basándose en las *wikis* a la que han aportado los usuarios. Dada una wiki se consultó que usuarios aportaron a esa wiki, para luego buscar por

cada usuario a que otras *wikis* han aportado y de esta manera obtener para esa *wiki* una lista de posibles *wikis* a recomendar.

Para tratar en lo posible de no perder la consistencia de la información entre la wiki que el usuario consultó y las wikis recomendadas se aplicó el coeficiente de similitud de Jaccard entre la wiki consultada y cada una de las posibles wikis a recomendar. Este coeficiente nos permite obtener un valor que nos indica el porcentaje de similitud que hay entre dos *wikis* basándose en el número de usuarios que han aportado tanto a una o ambas *wikis*, al final son presentadas al usuario aquellas *wikis* que obtuvieron el porcentaje de similitud más elevado.

Podemos decir que el resultado de las recomendaciones para una wiki muestra otros tópicos que también gustan a los usuarios que aportaron con dicha wiki, pero que no necesariamente traten sobre la misma temática.

INDICE GENERAL

ABREVIATURAS

API	Application Programming Interface
JVM	Java Virtual Machine
SIM	Similitud
XML	Extensible Markup Language

INDICE DE GRÁFICOS

INDICE DE TABLAS

INTRODUCCIÓN

La Wikipedia es actualmente la enciclopedia libre disponible en la Web con mayor aceptación entre los usuarios de todo el mundo. Su popularidad se debe, en parte, a su rápido y constante crecimiento, en cuanto a contenidos se refiere.

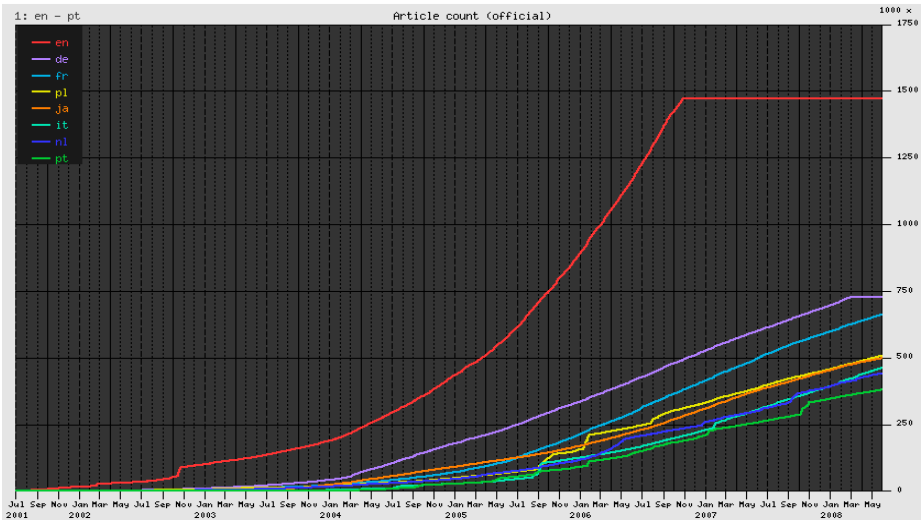

Figura 2. Crecimiento de la Wikipedia (Julio 2001 – Mayo 2008) [26]

Así como crece la Wikipedia, también crece la información disponible en otros sitios Web, como Facebook, YouTube, y Flickr. Con un volumen de información que crece de forma desmedida día a día, los *centros de datos* empiezan a ganar protagonismo junto a los *sistemas de recomendaciones* para evaluar y filtrar la gran cantidad de información disponible en la Web y guiar a los usuarios a la información que buscan.

Para estas empresas lograr hacerlo con un bajo presupuesto sin perder características indispensables como la tolerancia a fallos, alta disponibilidad, recuperación de fallos,

consistencia en los datos, *escalabilidad* y seguridad es uno de los problemas a los que se enfrentan día a día. Es aquí donde la plataforma Apache Hadoop [2] entra a ganar mercado, sobre todo en cuanto al uso del *paradigma* de *programación distribuida* popularizado por Google llamado *Map/Reduce* [3], que esta plataforma implementa de manera libre.

Al tener herramientas que nos permiten manejar grandes cantidades de información la forma de presentarlas a los usuarios al momento de realizar búsquedas es otra meta de nuestros días, y es aquí donde los sistemas de recomendación juegan un papel crucial. La idea tras ellos es encontrar usuarios con gustos similares a los de otro determinado y recomendar a este cosas que desconoce pero que gustan a aquellos con los que se tiene similitud.

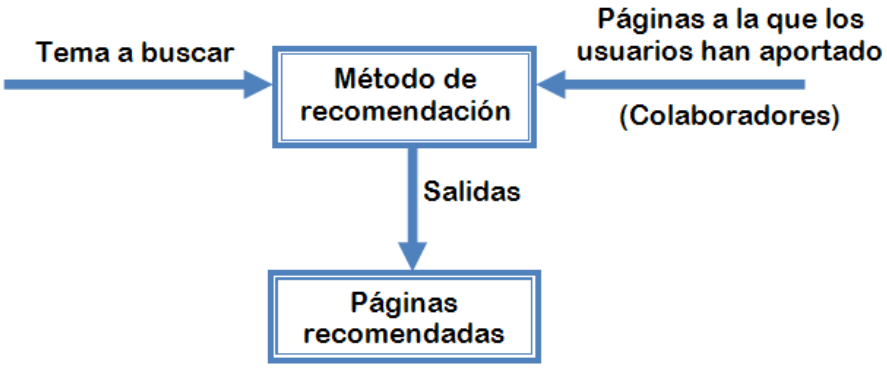

Figura 3. Estructura de un Sistemas de recomendaciones. [27]

El gráfico de la figura 3 muestra una estructura simple de un sistema de Recomendación adaptado específicamente a nuestro proyecto.

El tema central del presente trabajo es brindar una panorámica general del uso y aplicación del *paradigma Map/Reduce* y de los sistemas de recomendación, así como resultados obtenidos

empleando el coeficiente de similitud de Jaccard como parte del método de recomendación para generar nuestras salidas. El campo específico de aplicación de estos conceptos es en la generación de recomendaciones para la Wikipedia, en base a las contribuciones de sus usuarios.

1. ANÁLISIS CONTEXTUAL

1.1. SOBRE LA WIKIPEDIA

En la actualidad, la Wikipedia es la enciclopedia libre mas consultada en Internet, la cual cuenta con más de 13,7 millones de artículos redactados por voluntarios de todo el mundo [4] .La Tabla 1 presenta el número de ediciones para las 10 mayores Wikipedias [5].

La *Fundación Wikimedia* [6] periódicamente realiza copias de seguridad de todos los contenidos de las diferentes bases de datos de Wikipedia como: Wikisource, Wikiquote, Wikcionario, etc. A estas copias se las conoce como *dumps*.

Estas copias están disponibles para toda la comunidad de desarrolladores en formato *XML*, y contienen información sobre el historial de aportaciones de usuarios a las páginas, título, revisiones, texto de la página, id de la página etc.

Idioma	#Editores registrados	#Revisiones	#Bots
EN	1,824,439	167,464,014	388
DE	226,912	37,367,801	169
FR	127,767	25,821,354	151
PL	51,796	10,465,003	100
JA	90,828	17,524,766	57
NL	60,749	10,691,679	103
IT	62,690	12,798,068	158
PT	64,994	8,904,662	69
ES	132,239	14,198,257	122
SV	26,972	5,583,020	93

Tabla 1. Parámetros descriptivos adicionales para las 10 mayores Wikipedias

Al ser la Wikipedia una gran fuente de información estructurada, y al incluir información sobre las aportaciones que los editores de la misma hacen a las diferentes páginas, resulta una fuente de información ideal como base para desarrollar un módulo de recomendaciones en base a aportaciones de usuarios, utilizando técnicas actuales de procesamiento masivo de datos.

Dicho módulo puede ser implementado en otras *Wikis*, con pocas modificaciones y en otros sitios que permitan aportaciones de una gran base de usuarios.

1.2. OBJETIVOS DEL PROYECTO

1.2.1. Objetivo General

Desarrollar un módulo de recomendaciones para la Wikipedia basado en las *wikis* a la que los usuarios aportan, utilizando el *paradigma* MapReduce y herramientas libres para el procesamiento masivo de datos.

1.2.2. Objetivos específicos

- Analizar los *dumps* de la Wikipedia en español proporcionados por *La Fundación Wikimedia* utilizando Hadoop.
- Obtener para cada página de la Wikipedia un conjunto de posibles páginas a recomendar, consultando en que otras páginas escribieron los usuarios aportantes de dicha wiki.
- Implementar el coeficiente de similitud de Jaccard en su versión Map/Reduce para seleccionar las mejores páginas a recomendar.
- Verificar si los usuarios de la Wikipedia en general, aportan sobre temas similares de parecido contenido o al contrario las aportaciones de los usuarios es arbitraria y no sigue una misma temática.
- Analizar la escalabilidad de los diferentes procesos Map/Reduce utilizados en el desarrollo del proyecto.

2. MARCO TEÓRICO

2.1. HERRAMIENTAS PARA EL PROCESAMIENTO MASIVO Y ESCALABLE DE DATOS

El procesamiento masivo de datos, usando Hadoop como plataforma libre de procesamiento escalable y distribuido parte del uso y compresión de su estructura y elementos que participan tanto en su diseño como implementación, a continuación veremos unos aspectos importantes a tener en cuenta.

2.1.1. MAP/REDUCE [8]

Map/Reduce es un *framework* introducido por Google para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadoras.

Las funciones Map y Reduce están definidas ambas con respecto a datos estructurados en pares (clave, valor). Map toma uno de estos pares de datos con un tipo en un dominio de datos,

y devuelve una lista de pares en un dominio diferente:

Map (k1, v1) -> list (K2, v2)

La función de mapeo es aplicada en paralelo para cada ítem en la entrada de datos. Esto produce una lista de pares (k2, v2) por cada llamada. Después de eso, 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.

La función reduce es aplicada en paralelo para cada grupo, produciendo una colección de valores para cada dominio:

Reduce (K2, list (v2)) -> list (v2)

Cada llamada a Reduce típicamente produce un valor v2 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 Map/Reduce* transforma una lista de pares (clave, valor) en una lista de valores. Este comportamiento es diferente de la combinación "map and reduce" de programación funcional, que acepta una lista arbitraria de valores y devuelve un valor único que combina todos los valores devueltos por mapa.

2.1.2. HADOOP DISTRIBUTED FILE SYSTEM (HDFS) [8]

HDFS es un sistema de archivos distribuido que sigue una *arquitectura cliente/servidor* basada en un único nodo servidor denominado Nameserver que maneja el Namespace del sistema de ficheros y regula el acceso a los mismos, y varios clientes llamados *Datanodes* que gestionan el almacenamiento de los nodos que contienen.

Internamente, un fichero es dividido en bloques, y estos bloques son almacenados en un conjunto de *Datanodes*. Todos los bloques excepto el último son del mismo tamaño (64Mbs por defecto).

Los bloques de un fichero son replicados, con el fin de dotar al sistema de tolerancia a fallos. El tamaño de bloque y el número de réplicas es parametrizable para cada fichero. La optimización de la colocación de las réplicas en los diferentes *Datanodes* es uno de los factores que distingue a este sistema de ficheros.

Soporta la típica estructura jerárquica de directorios, con posibilidades de crear, borrar, mover y renombrar tanto directorios como ficheros. Los ficheros borrados, serán en realidad movidos temporalmente a una ‘papelera’ localizada en /trash.

Es accesible de manera nativa mediante un *API* para java y se puede navegar por su contenido usando un navegador Web.

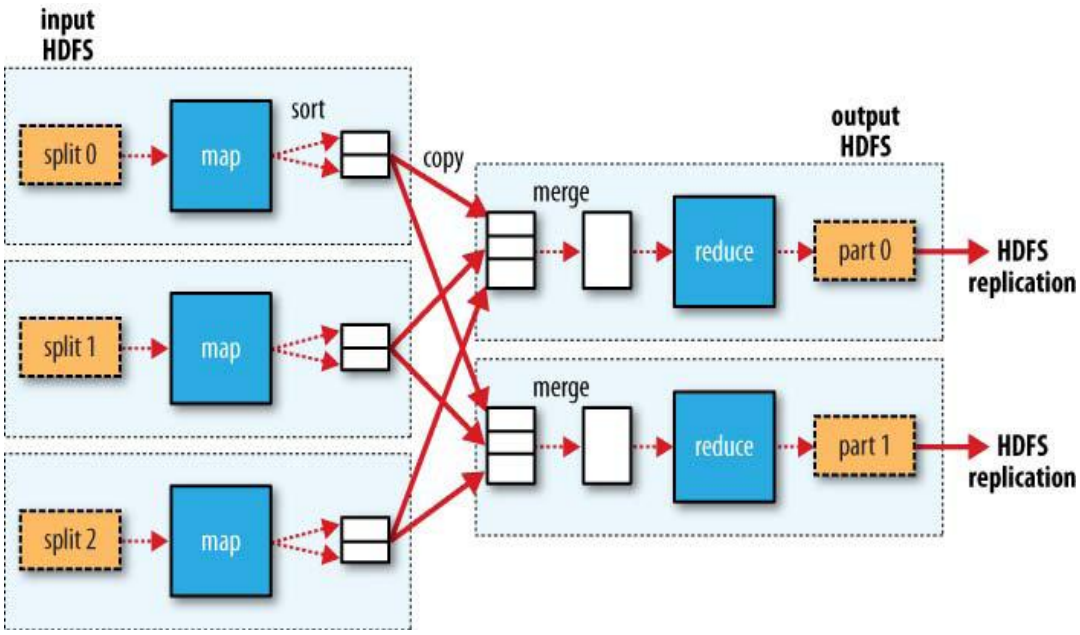

Figura 4. Flujo de datos Map/Reduce con dos tareas reduce, los datos de entrada y la salida se encuentran en el HDFS [28]

2.1.3. AMAZON ELASTIC MAPREDUCE

Amazon Elastic MapReduce [9] es un servicio Web que permite a las empresas, investigadores, analistas de datos, y desarrolladores de forma fácil y rentable procesar grandes cantidades de datos. Se utiliza un marco organizado Hadoop se ejecutan en el Web de infraestructura de *Amazon Elastic Compute Cloud (Amazon EC2)* [10] y *Amazon Simple Storage Service (Amazon S3)* [11].

Amazon Elastic MapReduce automáticamente incrusta una implementación del *framework* MapReduce de Hadoop en las instancias de Amazon EC2, sub-dividiendo los datos de un flujo de trabajo en pequeñas partes, de forma que ellos puedan ser procesados (la función "map") en

paralelo y, eventualmente, re combinando los datos en una solución final (la función "reduce").

Amazon S3 sirve como fuente para los datos que se están analizando, así también como el destino para el resultado final.

Precisamente Elastic MapReduce se centra en el concepto de un flujo de trabajo. Cada Flujo de trabajo puede contener uno o más pasos.

Cada paso recibe un paquete de datos de Amazon S3, distribuye los datos a un determinado número de instancias de EC2 que están funcionando con Hadoop (se aumentan las instancias si es necesario), se hace todo el trabajo y, a continuación, escribe los resultados nuevamente en S3. Cada paso debe hacer referencia a un código "mapper" o un "reducer" específico de la aplicación (JAR o código de script para uso vía streaming model).

Amazon EC2 y Amazon S3 son facturados por separado. La fijación de precios para Amazon Elastic MapReduce es por instancia-hora consumida para cada tipo de instancia, desde que el flujo de trabajo comenzó a tramitar hasta que se pone término al mismo. Cada instancia parcial de horas consumidas se cobra como una hora completa.

Con S3, se paga tan sólo lo que se utiliza. En dos rubros diferentes: por espacio que consumes al mes, y por peticiones que recibe el servidor. Los datos transferidos entre Amazon EC2 y Amazon S3 están libres de cargos (es decir 0,00 dólares por GB).

Instancia sobre demanda	Amazon Ec2 Precio por hora	Amazon Elastic MapReduce Precio por hora
Small (default)	$0.10	$0.015
Medium	$0.20	$0.03
Large	$0.40	$0.06
Extra Large	$0.80	$0.12

Tabla 2. Precios de Amazon EC2 y Amazon Elastic MapReduce Actualizados a Septiembre del 2009 [12]

En el desarrollo del proyecto utilizamos Instancias Medium de Amazon Elastic MapReduce.

ALMACENAMIENTO	TRANSFERENCIA DE DATOS
$0.150 por GB – primeros 50 TB / mensuales de almacenamiento utilizado	$0.100 por GB – Toda transferencia de entrada
$0.140 por GB – siguientes 50 TB / mensuales de almacenamiento utilizado	$0.170 por GB – primeros 10 TB / mensuales de transferencia de salida
$0.130 por GB – siguientes 400 TB / mensuales de almacenamiento utilizado	$0.130 por GB – siguientes 40 TB / mensuales de transferencia de salida
$0.120 por GB – Almacenamiento / mensuales después de 500 TB	$0.110 por GB – siguientes 100 TB / mensuales de transferencia de salida

Tabla 3. Precios de Amazon s3, actualizados a Septiembre del 2009 [13]

3. MATERIALES

3.1. DATASET

Como *Dataset* elegimos trabajar con los respaldos de la Wikipedia, todo el contenido de la Wikipedia se encuentra bajo licencia *GNU* de documentación libre, por lo tanto puede ser copiado, modificado y redistribuido. La *Fundación Wikimedia* es la encargada de distribuir periódicamente copias de seguridad de todos los contenidos de las diferentes bases de datos de Wikipedia y a estos se los llama *dumps*.

Todo este contenido puede descargarse desde http://download.wikimedia.org/. Estos *dumps* están disponibles en formato *XML* y comprimidos con bzip2.

El contenido de las páginas se encuentra entre los *tags* <page></page> y las revisiones entre los *tags* <revisión></revisión> el formato de estos archivos *XML* es de la siguiente forma.

Figura 5. Formato de los Dumps de Wikipedia

En la página de descargas, hay varios ficheros para descarga, pero los más importantes para nuestro proyecto son:

Pages-articles.xml.bz2 que contienen las últimas revisiones de los artículos, este archivo es un poco pesado debido a que guarda todo el contenido de la página.

pages-stub-meta-history.xml.7z/bz2 que solo ofrece información sobre el historial de las modificaciones y guarda el usuario que realizó dicha modificación.

Los archivos que decidimos utilizar para realizar las pruebas fueron los siguientes: **eswiki-20090710-pages-articles.xml** (3.18 GB) y **eswiki-20090710-stub-meta-history.xml** (7,04 GB) es decir los respaldos del 10 de Julio del 2009 de la edición de la Wikipedia en español.

El archivo eswiki-20090710-stub-meta-history.xml fue el *dataset* utilizado para realizar todos los procesos *Map/Reduce* necesarios para obtener las recomendaciones mientras que el

archivo eswiki-20090710-pages-articles.xml se lo utilizó para importar a la base de datos del MediaWiki todo el contenido de la Wikipedia y poder verla offline. El MediaWiki es un software de wiki libre del cual hablaremos más adelante, esta importación se la realiza con un programa hecho en java llamado **mwdumper.jar** [14] que es gratuito y está disponible en Internet.

3.2. SOFTWARE WIKI (MEDIAWIKI)

MediaWiki [7] es un software wiki libre escrito originalmente para Wikipedia, actualmente es utilizado por otros proyectos *wikis* de la *Fundación Wikimedia* y también por otras *wikis*, incluyendo la wiki de la ESPOL. Se lo utilizó en nuestro proyecto como software para visualizar las *wikis* y resultados de las recomendaciones.

MediaWiki está desarrollado en PHP y puede ser usado con una base de datos *MySql* o *PostgreSQL*. La versión del MediaWiki utilizada por nosotros es la 1.15.0 y puede descargarse desde la siguiente dirección:
http://download.wikimedia.org/mediawiki/1.15/mediawiki-1.15.0.tar.gz

La principal ventaja del MediaWiki es que al ser licencia *GNU*, este software ofrece todo el *código fuente* de tal manera que se lo puede modificar y ajustar a los requerimientos de nuestro proyecto.

3.3. CLOUD[9]

Cloud[9] es una librería MapReduce para Hadoop, de código abierto que se encuentra en un *repositorio Subversión*, para obtenerlo se lo realizó desde *Eclipse* abriendo la perspectiva *SVN* Repository Exploring y añadiendo el siguiente repositorio:

https://subversion.umiacs.umd.edu/umd-hadoop/core

Toda esta y más información del *Cloud9* como el *API* Javadoc la obtuvimos desde la Web del autor, Jimmy Lin [15].

Utilizamos esta librería ya que entre sus varias opciones ofrece un paquete que permite trabajar con los *dumps* de la Wikipedia. Este paquete es edu.umd.*cloud9*.collection.wikipedia. Entre las clases que contiene este paquete y la que utilizamos con mayor frecuencia es *WikipediaPage* que representa una página Wikipedia (contenido que se encuentra entre los *tags* <page></page>) y nos permite obtener–entre otras–cosas el título de la página, el id y el texto *XML* de la página en sí. Utilizamos *WikipediaPage* como dato de entrada en algunos Mapper del proyecto.

Otra clase importante es la que nos permite indicar en el *Job* que el formato de entrada sea de tipo WikipediaPageInputFormat, de tal manera que enviando al *Job* como entrada un solo archivo *XML* donde están todas las páginas de la Wikipedia. Esta librería se encarga de enviarnos al map una sola página para ser procesada.

3.4. HADOOP

Hadoop [16] es una plataforma que nos permite desarrollar aplicaciones que tengan que tratar con grandes cantidades de datos. Es un subproyecto de Apache open – source que implementa en Java algunas de las tecnologías de Google, como MapReduce y Google File System.

En Hadoop tenemos en lugar de *GFS* (Google File System) [3] el *HDFS* que es el sistema de archivos distribuidos pensado para grandes tamaños (GBs a TBs), *tolerante a fallos* y diseñado para ser instalado en máquinas de bajo coste que puedan ejecutar una instancia de la *JVM*.

Nosotros en nuestro proyecto utilizamos la versión 0.18.3 de Hadoop [17].

Todo el proceso de instalación de Hadoop lo evitamos ya que para correr nuestros programas lo realizamos sobre los Amazon Web Services (EC2, S3).

Utilizamos el servicio EC2 con AMIs (Amazon Machine Image) [18] que ya traen instalado Hadoop sobre un *sistema operativo Fedora*. S3 lo utilizamos como repositorio para los datos de entrada y para la salida final de los procesos MapReduce.

4. DISEÑO Y METODOLOGÍA UTILIZADA

Para realizar el módulo de recomendaciones para la Wikipedia hemos dividido el proceso en dos partes. La primera se encarga del filtrado de información para luego proceder a la valoración de los datos y por último la selección de los datos que consiste en clasificar y ordenar los datos obtenidos en el proceso anterior.

4.1. FILTRADO DE INFORMACIÓN

Teniendo como hipótesis que los usuarios de la Wikipedia la mayoría de las veces opina sobre temas similares, el primer paso consiste en obtener los usuarios que han contribuido con las ediciones de una página para luego recuperar por cada usuario las páginas a las que ellos también han hecho aportaciones.

Entre los diferentes tipos de usuario que frecuentan la Wikipedia encontramos los usuarios anónimos y los usuarios registrados.

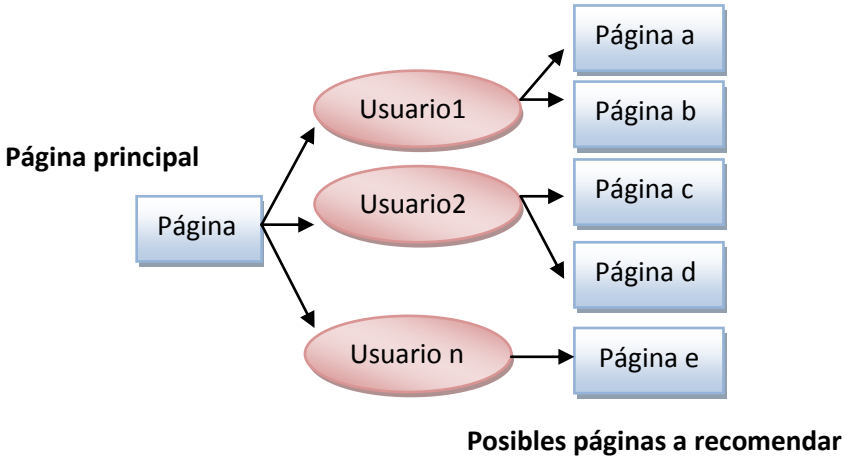

Figura 6. Filtrado de información

Los Usuarios anónimos son aquellos que no se han registrado y la Wikipedia en sus revisiones los identifica con su dirección IP, mientras que los usuarios registrados son quienes han seguido un proceso muy simple en el que solo es necesario dar el nombre y contraseña.

La Wikipedia cuenta con «*programas robot*» *(bots)* que detectan cambios de tipo vandálico y que actúan en consecuencia, a veces de forma automática, a veces preventiva. Por ejemplo si en un artículo aparecen insultos estos bots se encargan de eliminarlos, estos cambios también son almacenados en las revisiones de la Wikipedia. Una forma sencilla de identificar si un usuario es un robot es verificando si contiene la palabra bot en su nombre de usuario; por ejemplo, el robot principal de la Wikipedia se llama AVBOT [19].

Durante el proceso de filtrado los usuarios anónimos y los robot no son considerados, debido a que estos no siguen una línea de aportaciones sobre un tema definido y su inclusión traería como resultados páginas muy aleatorias.

También son descartadas las páginas que no corresponden a un artículo, con el fin de reducir el número de páginas a procesar. La Wikipedia entre sus tipos de páginas tiene las páginas de discusión, redirects (páginas que se encargan de direccionar a otras páginas), páginas de configuración de la Wikipedia y artículos.

Para comprobar si una página no corresponde a un artículo se verificó si contiene el signo dos puntos (:) en su título; por ejemplo, la página con título Wikipedia Discusión: Café (noticias) no corresponde a un artículo.

Este proceso no es de vital importancia, al no intervenir en la obtención del resultado final; sin embargo, lo describimos ya que fue el que nos dio la pauta para comenzar la investigación, entender la estructura de la Wikipedia, comprender cómo se comporta Hadoop con grandes datos de entrada y verificar que la hipótesis planteada por nosotros acerca de que los usuarios aportan generalmente a temas en común es valedera aunque no en un 100% ya que se observó que en algunos casos las posibles páginas a recomendar no se alejaban mucho del tema principal.

Observando esta primera salida pudimos conocer la magnitud de los datos a procesar, darnos cuenta de qué datos dentro del *XML* servían, y qué datos no; todo esto con el objetivo de reducir los tamaños de la salida para optimizar los procesos.

4.2. VALORACIÓN DE PÁGINAS A RECOMENDAR

Para valorar qué páginas son las mejores páginas a recomendar para la página principal, hemos decidido utilizar el coeficiente de similitud de Jaccard [20].

4.2.1. COEFICIENTE DE SIMILITUD DE JACCARD

También conocido como el índice de Jaccard, el coeficiente de similitud de Jaccard es una medida estadística de la similitud entre los conjuntos de la muestra.

Para los dos conjuntos, esto es definido como la *cardinalidad* de su intersección dividido por la *cardinalidad* de su unión, matemáticamente es expresada de la siguiente manera.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Para el caso de la Wikipedia es necesario un enfoque ligeramente diferente. Tomando como conjunto los usuarios de la Wikipedia que han aportado a dos páginas específicas, es posible determinar la similitud que hay entre estas dos páginas.

Para el cálculo de la similitud de pares de páginas, el numerador es el número de usuarios que han editando ambas páginas mientras que el denominador es el número de usuarios que han editado una o las dos páginas. Matemáticamente, tenemos la siguiente fórmula.

$$J(A,B) = \frac{|X \cap Y|}{|X \cup Y|}$$

Donde A y B son las páginas a las que se desea encontrar la similitud, X y Y corresponden al conjunto de usuarios que se producen con A y B respectivamente.

Gráficamente podemos verlo de la siguiente manera:

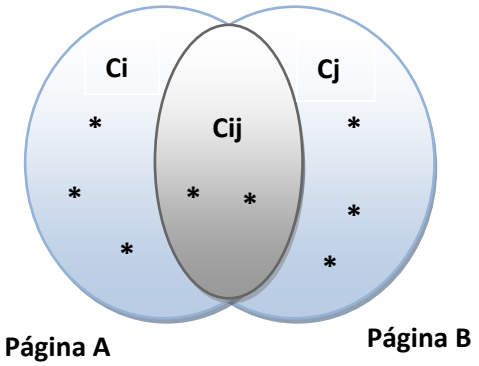

Figura 7. Diagrama de Venn para dos páginas de la Wikipedia con sus usuarios

Tomando en cuenta que Ci es el número de usuarios que han aportado a la página A, Cj es el número de usuarios que han aportado a la página B y Cij es el número de usuarios que han aportado tanto a la página A como a la página B, tenemos que la similitud entre estas dos páginas (Coeficiente de similitud de Jaccard) se calcula de la siguiente manera:

$$Sim(A,B) = \frac{\text{Cij}}{\text{Ci} + \text{Cj} - \text{Cij}}$$

Sim(A, B) es un *valor porcentual* y va desde 0 hasta 1 siendo este el valor más alto de coincidencia.

En el caso de que no exista ningún usuario que haya aportado tanto a la página A como a la página B, Cij valdría cero y el resultado del la Sim(A,B) será cero; es decir, que es probable que estas dos páginas no tengan absolutamente nada en común.

Otro caso extremo se produce cuando Ci y Cj son cero y todos los usuarios tanto de la página A como de la página B han aportado a ambas páginas; en este caso, la Sim (A, B) será 1, es decir es probable que estas dos páginas tengan mucho en común.

4.3. SELECCIÓN DE PÁGINAS A RECOMENDAR

En el proceso de valoración de datos obtuvimos todas las combinaciones posibles de las páginas a recomendar con sus respectivos valores de similitud. En esta etapa se toma como entrada estas combinaciones y se genera para cada página una lista de páginas a recomendar ordenadas de mayor a menor dependiendo del valor de similitud que tengan con respecto a la página siendo analizada. La Figura 8 presenta nuestra salida final, lista para ser ingresada a la base de datos *MySql*.

(Pag1, Pág. a)	0.20
(Pag1, Pág. b)	0.40
(Pag1, Pág. c)	0.70
(Pág. a, Pág. b)	0.10
(Pág. a, Pág. c)	0.01
(Pág. b, Pág. c)	0.03

Figura 8. Formato de salida del proceso de valoración de datos

Para elegir las 5 mejores páginas a recomendar, se seleccionan las páginas que tengan la

similitud (coeficiente de similitud de Jaccard) más cercano a uno.

Por ejemplo

Sim (Pag1, Pág. a) = 0.2
Sim (Pag1, Pág. b) = 0.4
Sim (Pag1, Pág. c) = 0.7

Entonces tenemos que para la Pag1 la mejor página a recomendar sería la Pág. C

Pag1	Pág. c, 0.7 ^Pág. b, 0.4 ^Pág. a, 0.2
Pág. a	Pag1, 0.2 ^Pág. b, 0.10 ^Pág. c, 0.01
Pág. b	Pag1, 0.4 ^Pág. a, 0.10 ^Pág. c, 0.03

Figura 9. Formato de salida final

4.4. ALGORITMO UTILIZADO

Para la implementación del algoritmo se utiliza la metodología *Map/Reduce*.

4.4.1. ALGORITMO UTILIZADO PARA EL FILTRADO DE INFORMACIÓN

Para obtener el universo de posibles páginas a recomendar se necesitaron de dos procesos *Map/Reduce*, primero hay que obtener un listado de páginas por usuario, luego por cada página de esa lista se genera una nueva lista de páginas. Al final se concatenan todas las listas de determinada página y se obtiene como resultado un archivo donde cada línea contiene una página y una lista de posibles páginas a recomendar. Los procesos Map y Reduce se detallan a continuación (ver Figura 10).

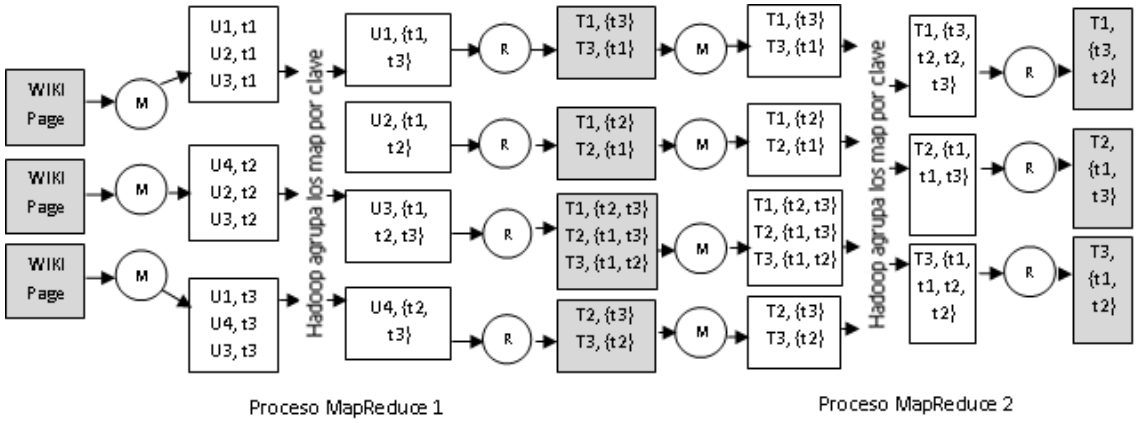

Figura 10. Procesos Map/Reduce para el Filtrado de Información

4.4.1.1. PROCESOS MAP/REDUCE PARA FILTRADO DE INFORMACIÓN

Primer Proceso *Map/Reduce*

Map

El método Map consiste generar *tuplas* <usuarios, página>, teniendo como dato de entrada las páginas de la Wikipedia con la información de las revisiones en formato *XML*.

La librería *Cloud*[9] nos permite procesar el *XML* de la Wikipedia y tener como dato de entrada un objeto de tipo *WikipediaPage*.

La clase *WikipediaPage* nos proporciona ya gran parte de la información de la página como el título, el id, el tipo de página, el *XML* en sí de la página, el texto, entre otros.

Entrada: *WikipediaPage*

Salida: Text, Text

WikipediaPage → <Usuario, Título Página>

Reduce

El reduce recibe una lista de páginas y se encarga de generar otra lista de páginas por cada una de las páginas de la lista recibida. Esta nueva lista generada es la misma lista recibida pero sin tomar en cuenta la página a la que le está generando la lista, ya que esta página es la clave.

Por ejemplo, un reduce recibió los siguientes valores:

Usuario1, {pág. 1, pág. 2, pág. 3, pág. 4}

Se generó una salida por cada página de la siguiente manera:

pág. 1, {pág. 2, pág. 3, pág. 4}
pág. 2, {pág. 1, pág. 3, pág. 4}
pág. 3, {pág. 1, pág. 2, pág. 4}
pág. 4, {pág. 1, pág. 2, pág. 3}

Entrada: Text, {Text}

Salida: Text, Text

Usuario, {Título Página} → Título Página, {Título Página}

Segundo Proceso *Map/Reduce*

Map

El método Map recibe una línea del archivo generado en el proceso MapReduce1. Esta línea tiene la forma [Título Página, {Título Página}] y simplemente lo que hace es emitir un map por

cada línea del archivo.

Entrada: Text, Text

Salida: Text, Text

Título Página, {Título Página} → <Título Página, {Título Página}>

Reduce

El reduce se encarga de concatenar la lista de páginas recibidas y este sería el universo de posibles páginas a recomendar.

Entrada: Text, {Text}

Salida: Text, Text

Título Página, {Título Página}→ Título Página, {Título Página}

4.4.2. ALGORITMO UTILIZADO PARA OBTENER EL COEFICIENTE DE SIMILITUD DE JACCARD

Para calcular el coeficiente de similitud de Jaccard para un par de elementos X, es necesario el cálculo de dos valores. El primero es el número de elementos de Y que se produce con los dos elementos X, y el segundo es el número de elementos de Y que se producen con uno o ambos elementos X.

Para el cálculo de la segunda cantidad, primero hay que calcular el número de elementos de Y que se produce con el elemento de primer X más el número de elementos de Y que se produce

con el segundo elemento X, y luego restar el número de elementos de Y que se producen con ambos.

El valor de la primera cantidad es obtenido mientras se calcula la segunda cantidad y por último este valor se lo divide para la segunda cantidad obteniendo así el valor del coeficiente de similitud de Jaccard para un par de páginas.

La implementación se la realizó utilizando tres procesos *Map/Reduce* que se describen a continuación (ver Figura 11).

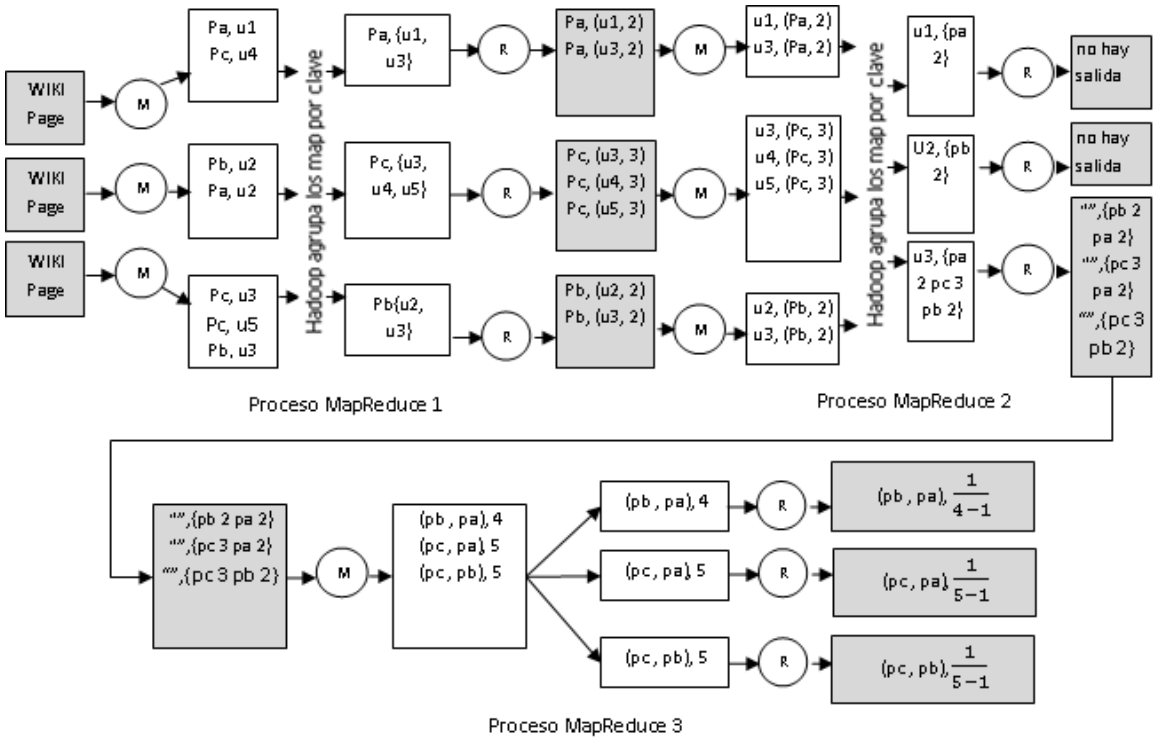

Figura 11. Procesos Map/Reduce para obtener el Coeficiente de similitud de Jaccard

4.4.2.1. PROCESOS MAP/REDUCE PARA OBTENER EL COEFICIENTE DE SIMILITUD DE JACCARD

Primer Proceso *Map/Reduce*

Map

Este método simplemente genera una tupla <clave, valor> utilizando como clave la página y como valor el usuario que realizó la aportación; teniendo como dato de entrada las páginas de la Wikipedia con la información de las revisiones en formato *XML*.

Entrada: *WikipediaPage*

Salida: Text, Text

WikipediaPage → <Título Página, Usuario>

Reduce

El método reduce recibe una página con la lista de usuarios que han aportado a esa página, y calcula la *cardinalidad* para esa página que no es más que el número de usuarios que aportaron a dicha página (a este valor le llamaremos C). Este valor de C será usado después para calcular el denominador del coeficiente de Jaccard.

El reduce retorna un par donde la clave es la página y el valor es una nueva tupla <Usuario, C> a la que le llamaremos CPar. Por cada usuario de la página se genera una salida de este tipo.

Entrada: Text, {Text}

Salida: Text, CPar

Título Página, {Usuario} → Título Página, <Usuario, C>;

Donde C = | {lista de Usuarios} |

Segundo Proceso *Map/Reduce*

Map

Aquí se tiene como dato de entrada la salida del Proceso *Map/Reduce* 1, cada línea de los archivos de salida de dicho proceso tiene la forma [Título Página Usuario C].

Por cada línea del archivo se generó un map donde la clave es el Usuario y la salida es una nueva tupla <Página, C>; en otras palabras, la salida de este método map es la salida del método reduce del proceso MapReduce1 pero con los valores de Página y Usuario intercambiados.

Entrada: Text, CPar

Salida: Text, CPar

Título Página, <Usuario, C> → Usuario, <Título Página, C>

Reduce

Este recibe una lista de *tuplas* <Título Página, C> por cada usuario, y la salida de este reduce es muy peculiar ya que por clave se utilizó un string vacío y por valor simplemente se concatena la lista de valores CPar recibidos, de tal manera que cada línea del archivo de salida es una lista

de valores CPar con la forma <Título Página, C>.

Entrada: Text, {CPar}

Salida: Text, {CPar}

Usuario, {<Título Página, C>} → R, {<Título Página, C>}; Donde R = “” (String vacío)

Tercer Proceso *Map/Reduce*

Map

En este método se recibe la salida del proceso *Map/Reduce* 2 cada línea del archivo de salida tiene una lista de *tuplas* de la forma <Título Página, C>. Este proceso es el encargado de generar las *tuplas* de página <Xa, Xb> con el primer elemento de la lista y cada uno de los siguientes elementos, es decir una lista de tamaño n producirá n-1 *tuplas*, uno por cada elemento de la lista exceptuando el primero. Las *tuplas* generadas son la clave del map y el valor es la suma de los valores de C asociado a cada página.

Entrada: Text, {CPar}

Salida: Text, Intwritable

R, {<Título Página, C>} → <Xa, Xb>, Cij; Donde Cij = Ci + Cj

Reduce

Finalmente, este es el reduce que calcula el coeficiente de similitud de Jaccard.

Primero cuenta cuantas veces se generó la tupla <Xa, Xb> y este valor representa la

intersección de Xa con Xb, es decir el numerador del coeficiente de similitud de Jaccard. El valor del denominador se obtiene restando el valor de Cij asociado al par con el de la intersección.

Entrada: Text, {Intwritable}

Salida: Text, Floatwritable

<Xa, Xb>, {Cij} → <Xi, Xj>, $\frac{|\{Cij\}|}{Cij - |\{Cij\}|}$

4.4.3. ALGORITMO UTILIZADO PARA SELECCIONAR LAS PÁGINAS A RECOMENDAR

Este proceso es muy sencillo pero muy importante, ya que es el que nos permite obtener las recomendaciones para cada página ordenadas de mayor a menor dependiendo del valor del coeficiente de similitud de Jaccard. Consta de un solo proceso *Map/Reduce*. Se recibe como entrada la lista de todas las posibles combinaciones de páginas con su valor de similitud, y simplemente se emiten dos maps por cada combinación de página y en el Reduce se procede a ordenar y concatenar las páginas. Su proceso *Map/Reduce* se describe a continuación (ver Figura 12).

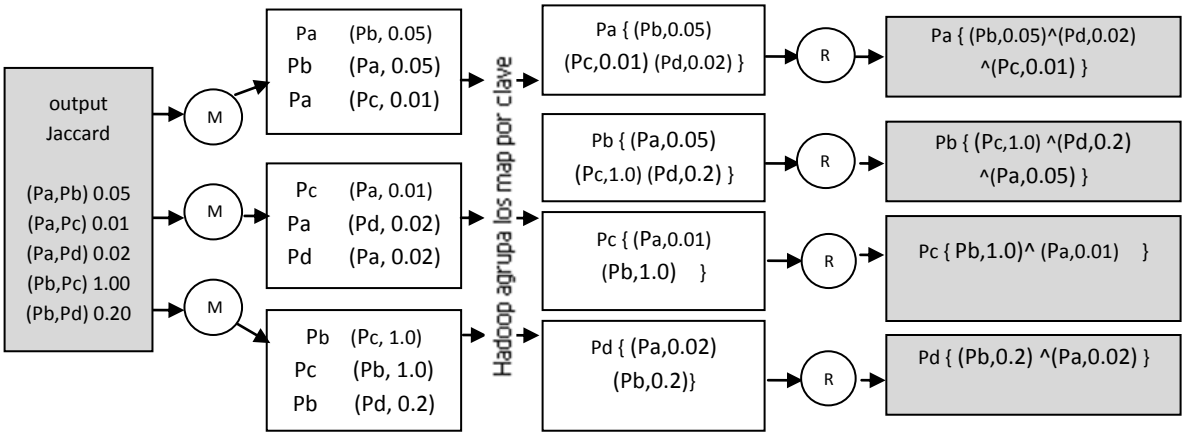

Figura 12. Proceso Map/Reduce para seleccionar las páginas a recomendar

4.4.3.1. PROCESO MAP/REDUCE PARA SELECCIONAR LAS PÁGINAS A RECOMENDAR

Map

Este método recibe una línea de la forma [Página A, Página B 0.2] que contiene un par de página con su respectivo valor de coeficiente de similitud entre ellas.

Se genera una tupla <clave, valor> utilizando como clave una página y como valor la otra página concatenada con el valor de similitud.

Como en el ejemplo la similitud entre la página A y la página B es 0.2 entonces se generan las dos siguientes salidas:

1. Una salida indica que la Página B es posible recomendación para la Página A con un grado de similitud de 0.2 <Página A, (Página B, 02)>.

2. La otra salida indica que la Página A es posible recomendación para la Página B con un

grado de similitud de 0.2 <Página B, (Página A, 02)>.

Entrada: Text, Text

Salida: Text, Text

[Título Página 1, Título Página 2, Valor Coeficiente] → <Título Página 1, (Título Página 2, Valor Coeficiente)> y <Título Página 2, (Título Página 1, Valor Coeficiente)>

Reduce

El reduce recibe una página con la lista de posibles páginas a recomendar asociadas con su valor de similitud. Este método se encarga de ordenar de mayor a menor esa lista dependiendo de su valor de similitud, se emite una salida con la página y la misma lista recibida pero ordenada.

Entrada: Text, {Text}

Salida: Text, {Text}

Título Página, {<Título Página, Coeficiente>} → Título Página, {<Título Página, Coeficiente>}

4.5. PRESENTACIÓN DE LOS RESULTADOS

Como se mencionó anteriormente se utilizó el MediaWiki para presentar los resultados. Este software utiliza una base de datos *MySql* para almacenar toda su información, la salida final de los diferentes procesos *Map/Reduce* es introducido a esta base de datos, ya que la salida es

simplemente un archivo de texto donde cada línea del archivo representa un registro, nosotros obtenemos todas las recomendaciones posibles para una página dependiendo del número de usuarios que ha aportado a esa página, pero solo son introducidos a la base de datos las 5 recomendaciones que tengan el índice de similitud de Jaccard más cercano a uno y estas son presentadas en la pestaña de recomendaciones. Esto con el objetivo de no saturar la base con datos innecesarios.

La tabla que almacena las recomendaciones es simple: contiene 6 campos, de los cuales, el primero corresponde a la página principal y los 5 restantes cada uno corresponde a una página recomendada.

Se modificó el MediaWiki y se le añadió para cada página una pestaña de recomendaciones que es donde se mostrarán los resultados (ver Figura 13).

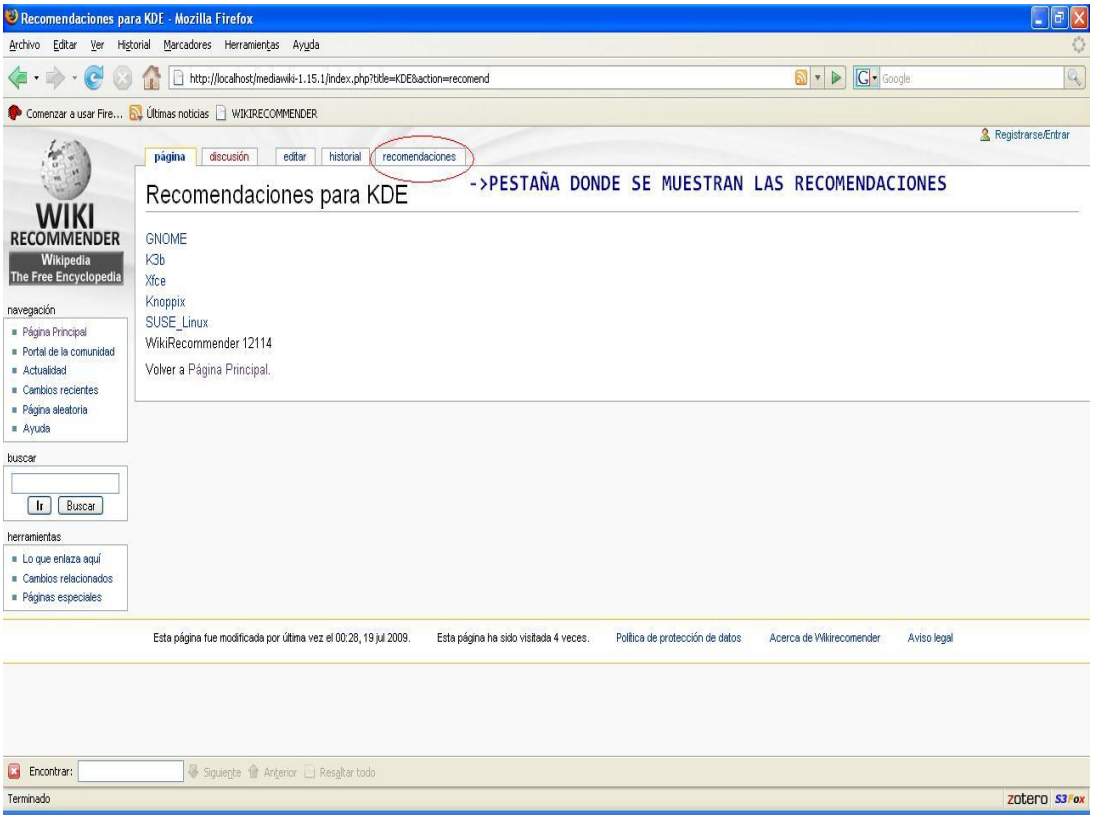

Figura 13. Interfaz del MediaWiki con la pestaña de recomendaciones

La arquitectura final del sistema es la siguiente (ver Figura 14).

Figura 14. Arquitectura final del sistema

Como se puede observar la arquitectura final del sistema es muy simple y en ningún momento se encuentra conexión con el EC2 o el S3 que son los servicios de Amazon que se utilizaron para generar las recomendaciones. Esto se debe a que el proceso de generar las recomendaciones no es un proceso en línea que se ejecuta cada vez que el usuario busca una página. Este es un proceso que se encuentra realizado con anterioridad y debería realizarse periódicamente dependiendo de cómo sea la tasa de cambio de los datos en la Wikipedia, o en su debido caso se lo puede realizar cada vez que se libere un nuevo respaldo de la base de datos de la Wikipedia.

5. HERRAMIENTAS DE DESARROLLO

Para implementar los algoritmos de los diferentes procesos *Map/Reduce* se usó el lenguaje de programación java (jdk1.6.0_07) con el IDE *Eclipse* 3.4.0 bajo un entorno de Windows XP SP2.

Para la implementación del *paradigma Map/Reduce* se utilizó la librería Hadoop 0.18.3

Además se utilizaron las siguientes librerías:

- Para utilizar y analizar los *dumps* de la Wikipedia como dato de entrada y poder procesar página por página se utilizó la librería *Cloud*[9].
- *Cloud*[9] no recupera todas las revisiones de una página por lo que para recuperar y analizar las revisiones a partir de un texto con formato *XML* se utilizó la librería JDOM 1.11 [21].

6. AMBIENTE DE PRUEBAS

Una vez implementado los diferentes procesos Map /Reduce era necesario probarlos, pero esta tarea no es tan sencilla ya que depende mucho de la cantidad de datos a procesar es por esto que la etapa se la realizó en dos ambientes.

El primer ambiente es utilizando el software VMWare Workstation 6.5.2 con una imagen de la distribución del *sistema operativo Ubuntu* con el kernel de Linux 2.6.27-11-server y entorno *GNOME*. Esta imagen ya trae tanto el Hadoop como el *HDFS* instalado y configurado; En este entorno probamos los algoritmos utilizando pequeños *dataset* de entrada que no sobrepasaron los 100Mb.

Para *datasets* más grandes como es el proporcionado por la Wikipedia que pesa 7,04 GB se utilizó de un *clúster* de 10 nodos para poder ejecutar nuestros procesos *Map/Reduce*; para estas pruebas utilizamos los servicios EC2 y S3 de Amazon.

6.1. EJECUTAR TRABAJOS MAP/REDUCE DESDE LA MÁQUINA VIRTUAL

Para correr un trabajo *Map/Reduce* hay tres pasos fundamentales: (1) generar el archivo .jar que contiene la implementación de nuestro proceso *Map/Reduce*, (2) subir el *dataset* utilizado por nuestro proceso al sistema de archivos distribuido utilizado por Hadoop (*HDFS*), y finalmente, (3) correr la clase principal llamada también Driver que se encuentra dentro del archivo .jar.

Ya sea en una ventana de terminal desde la *máquina virtual* de Ubuntu o conectado al *nodo maestro* de nuestro *clúster* vía *SSH*, las instrucciones son las descritas a continuación.

6.1.1. SUBIR UN ARCHIVO AL HDFS

hadoop dfs -copyFromLocal /eswiki-20090710-stub-meta-history.xml /entrada/ eswiki-20090710-stub-meta-history.xml

De esta manera subimos el archivo *XML* que se encuentra en el sistema de archivos local, al *HDFS* en la ubicación user/root/entrada/ eswiki-20090710-stub-meta-history.xml

6.1.2. EJECUTAR LA CLASE PRINCIPAL

hadoop jar archivo.jar paquete.claseprincipal /entrada /salida

Hay ocasiones en los que ya sea la entrada, la salida o ambas no será en el *HDFS* sino en el S3 que es el servicio de Amazon para repositorio de datos.

En este caso la ejecución del proceso será de la siguiente manera:

hadoop jar archivo.jar paquete.claseprincipal /entrada

s3n://ACCESS_KEY:PRIVATE_KEY@Bucket/salida

Notar que la entrada esta en el *HDFS* y la salida será en el S3 específicamente en la dirección Bucket/salida. Bucket es el nombre que se le da a las carpetas en la raíz de la cuenta S3.

Nosotros creamos un Bucket llamado WikiRecommender, el ACCESS_KEY y PRIVATE_KEY son las claves públicas y privadas respectivamente proporcionadas por Amazon al momento de contratar el servicio.

6.2. EJECUTAR TRABAJOS MAP/REDUCE UTILIZANDO LOS SERVICIO DE AMAZON EC2, S3

Antes de realizar las pruebas utilizando los servicios de Amazon realizamos los siguientes pasos

1. Subir al S3 el *dataset* que utilizaremos como dato de entrada, en nuestro caso eswiki-20090710-stub-meta-history.xml.gz y si es necesario el archivo .jar que contiene la implementación de nuestro programa *Map/Reduce*.
2. Levantar un *clúster* de N nodos sobre los que correrá nuestro proceso Map /Reduce.
3. Conectarse al *nodo maestro* vía *SSH*.
4. Descargar al *nodo maestro* los archivos que corresponden al *dataset* de entrada y el .jar que contiene nuestra aplicación.
5. Descomprimir el *dataset* de entrada y subirlo al Sistema de archivos distribuido que

utiliza Hadoop (*HDFS*).

6. Correr la clase principal llamada también Driver que se encuentra dentro del archivo .jar

6.2.1. SUBIR ARCHIVOS AL S3

Para subir archivos al S3 nosotros lo realizamos por medio del *plugin* de Firefox S3Fox [22], el cual se integra de manera muy intuitiva al Firefox y una vez instalado permite subir y descargar archivos (ver Figura 15), solo hay que configurar las credenciales de acceso (ACCESS_KEY y PRIVATE_KEY proporcionadas por Amazon) y listo.

El S3 permite subir un número ilimitado de archivos pero cada archivo no puede pesar más de 5 GB, es por esta razón que subimos el *dataset* comprimido en formato *Gzip* y no el *XML* directamente ya que el *XML* pesa (7,04 GB).

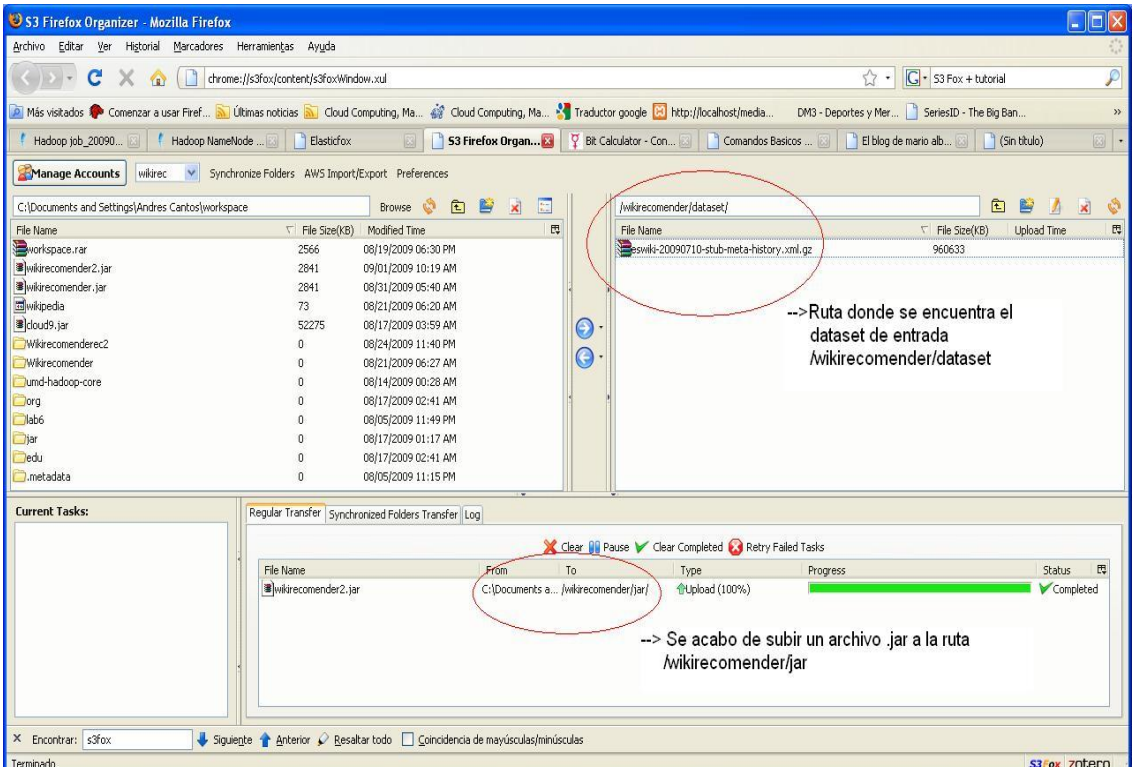

Figura 15. Ventana que muestra el plugin S3fox en Firefox

6.2.2. LEVANTAR UN CLÚSTER DE N NODOS EN EC2 DESDE LA MÁQUINA VIRTUAL Y CONECTARSE AL NODO MAESTRO

Para crear un *clúster* de N nodos desde la *máquina virtual*, primero abrimos una ventana de terminal y luego escribimos el siguiente comando:

cloudera-for-hadoop-on-ec2-0.4.0/bin/hadoop-ec2 launch-cluster nombreclúster N; nombre del *clúster* es el nombre que le daremos a nuestro grupo y N es el número de nodos que queremos que tenga nuestro *clúster*.

Una vez que está levantado el *nodo maestro* aparece un mensaje como el siguiente:

Master is ec2-75-101-199-41.compute-1.amazonaws.com; Donde 75-101-199-41 es la dirección IP del *nodo maestro*.

Podemos ver el estado de nuestro proceso MapReduce vía el puerto 50030 y podemos descargar los resultados generados por el programa vía el puerto 50070, ingresando desde un navegador la URL http://75.101.199.41:50030 ó http://75.101.199.41:500370 según sea el caso. Luego para conectarse al *nodo maestro* vía *SSH* se lo realiza con el siguiente comando:

ssh -i /home/training/.ec2/wikirecommender.pem

root@ec2-75-101-199-41.compute-1.amazonaws.com

Listo ya estamos conectados al *nodo maestro*, podemos verificar esto si el *prompt* del terminal cambio y tiene la forma **[root@ip-10-244-121-174 ~]** #; Donde 10-244-121-174 es la dirección IP interna que utiliza Amazon para identificar al *nodo maestro*.

6.2.3. DESCARGAR ARCHIVOS DEL S3 AL HDFS

Descargar archivos del S3 al *HDFS* es muy sencillo simplemente lo hacemos con la instrucción wget de la siguiente manera:

wget http://wikirecomender.s3.amazonaws.com/dataset/eswiki-20090710-stub-meta-history.xml.gz

Aquí nos descargamos el archivo eswiki-20090710-stub-meta-history.xml.gz que corresponde al *dataset* de entrada, notar que http://wikirecomender.s3.amazonaws.com corresponde a la dirección del Bucket WikiRecommender.

Luego para descomprimir el archivo y tener el *XML* utilizamos la siguiente instrucción:

gzip -d eswiki-20090710-stub-meta-history.xml.gz

Algo importante a tomar en cuenta es que el *nodo maestro* tiene 2 particiones, utilizando el comando df –h las podemos ver.

root@ip-10-244-121-174 ~] # df –h

File System	Size	Used	Avail	Use%	Mounted on
/dev/sda1	3.0G	1.6G	1.3G	56%	/
/dev/sda2	335G	7.3G	311G	3%	/mnt
none	874M	0	874M	0%	/dev/shm

Tabla 4. Particiones del nodo maestro

Nótese que la principal (/) tiene solo 3.0 G de espacio y la unidad montada sobre (/mnt) tiene 335 G de espacio por lo que el proceso de descarga y descompresión se lo realizó sobre esta unidad.

Para subir el archivo al *HDFS* y correr el programa *Map/Reduce* utilizamos las mismas instrucciones que se usaron en la *máquina virtual*.

7. PRUEBAS EXPERIMENTALES Y RESULTADOS

El objetivo de utilizar el *paradigma* MapReduce es probar el eficiente resultado y ahorro de tiempo que se obtiene sobre datos muy grandes, por lo que probar la *escalabilidad* del programa es muy importante. Estas pruebas consistieron en probar los procesos MapReduce desarrollados para nuestra investigación (*Proceso de filtrado*, Obtención del Coeficiente de Jaccard y proceso de selección de páginas) sobre rebanadas de datos de diferente tamaño, que van desde las 57.000 páginas hasta el millón de páginas siendo este un número cercano al número total de entradas que tiene la edición de la Wikipedia en español. Cada ensayo se realizó dos veces y los tiempos que se obtuvieron fueron promediados para realizar un gráfico tamaño de datos de entrada vs. Tiempo.

Las pruebas se realizaron bajo las siguientes condiciones:

- *Clúster* de 10 nodos médium
  Las características de cada nodo son las siguientes:
  - CPU Intel xeon - 2,33 Ghz
  - Memoria Ram 1,8 GB
  - Disco Duro 335 GB

- 10 tareas reduce para cada proceso MapReduce

7.1. PRUEBAS EN PROCESO DE FILTRADO

El proceso de filtrado internamente consta de dos procesos *Map/Reduce*, el primero se encarga de generar una lista de páginas por cada página de un usuario, mientras que el segundo se encarga de recibir esas listas y concatenarlas.

Los resultados obtenidos se muestran en la Tabla 5 y en la Figura 16.

# páginas entrada / tiempo en minutos que toma el proceso	57.000	112.468	216.903	459.866	1.064.418
1° Proceso MapReduce	4,52	8,38	20,45	39,70	84,1
2° Proceso MapReduce	15,80	43,10	46,18	135,22	331,55
Proceso Filtrado	20,32	51,48	66,63	174,92	415,65

Tabla 5. Tiempos del proceso de filtrado

En la Figura 16 se observa que ambos procesos *Map/Reduce* siguen una tendencia creciente y su *escalabilidad* es lineal, resultado que ya nos esperábamos, lo interesante es que el primer proceso no presenta cambios bruscos de tiempo al contrario de lo que sucede con el segundo proceso donde se observa que al duplicar el número de entradas el tiempo que toma terminar el proceso cambia bruscamente tanto así que se duplica y mientras crece la entrada hasta se triplica.

El segundo proceso *Map/Reduce* simplemente toma la lista de páginas obtenidas en el primer proceso y las concatena, eso involucra un tiempo dependiendo del número de elementos a

concatenar, pero a pesar de que esa programación no es nada compleja, se observa que es el proceso que toma más tiempo siendo así que es el que más influye en el tiempo total del *proceso de filtrado*, no sigue una tendencia evolutiva como el primer proceso *Map/Reduce*.

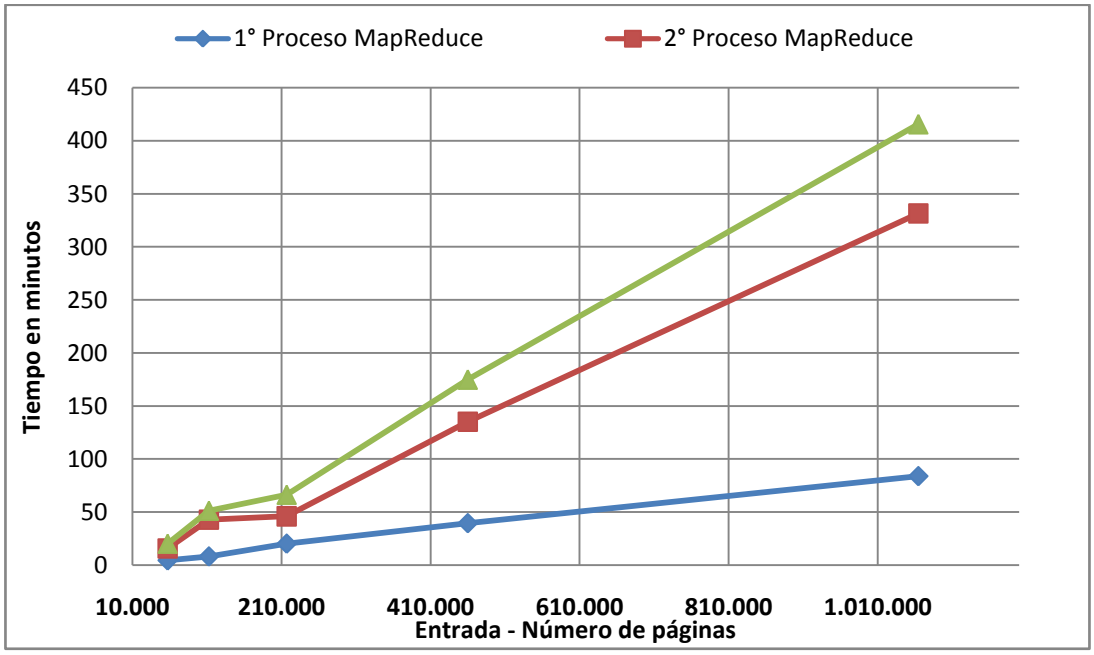

Figura 16. Resultado de pruebas para el proceso de filtrado

Esto se debe a que a mayor entrada de datos el tamaño de la salida del primer proceso lógicamente aumenta y la mayor parte del tiempo del segundo proceso *Map/Reduce* es el tiempo que demora Hadoop en leer los datos obtenidos del primer proceso *Map/Reduce*, emitir los map y agrupar estos map por clave.

Para ratificar esta conclusión se tomó el número de registros de salida del primer Proceso MapReduce variando el tamaño de la entrada, estos registros de salida corresponden al número

de líneas que serán la entrada para el segundo proceso MapReduce. Estos datos son proporcionados por Hadoop a través de la interfaz web por medio del puerto 50030.

Los resultados obtenidos se muestran en la Tabla 6 y en la Figura 17.

# páginas	57.000	112.468	216.903	459.866	1.064.418
# de Registros de salida	586.811	1.102.995	1.880.293	3.058.238	4.211.736

Tabla 6. Tamaño de la salida del primer proceso Map/Reduce

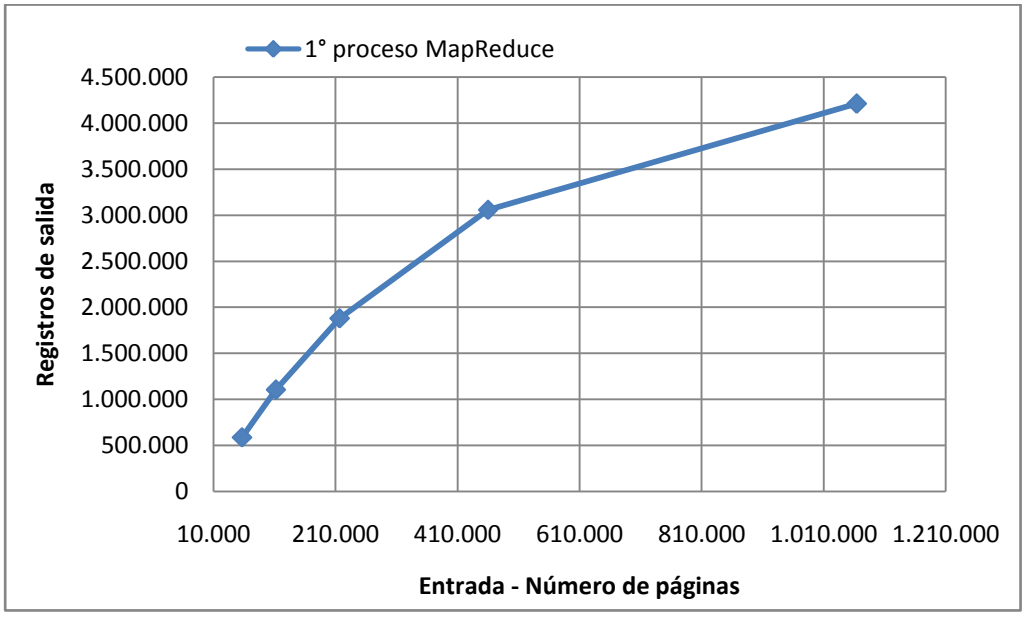

Figura 17. Tendencia del número de registros de salida variando el tamaño de la Entrada para el 1° Map/Reduce

Como se ve el número de los registros de salida para el primer MapReduce sigue una tendencia creciente.

Cabe destacar que al aumentar el número de páginas de entrada la diferencia entre una prueba y otra es de miles y hasta millones de registros.

Es por esta razón que el segundo proceso *Map/Reduce* presenta cambios bruscos en el tiempo de ejecución, porque por ejemplo para la prueba con 216.903 páginas Hadoop tiene que ordenar y agrupar 1.880.293 registros mientras que con 459.866 páginas tiene que ordenar y agrupar 3.058.238 es decir aumenta poco más de un millón de registro.

Hadoop para ordenar las salidas de los mapas y redirigirlas a los reduce, por defecto utiliza la clase HashPartitioner [8] que implementa la estrategia de particionamiento hash.

Las tablas hash, en el peor de los casos, pueden presentar tiempos de búsqueda de hasta O(n) [23] es decir depende del número de elementos, dependiendo de la aplicación se puede desarrollar y configurar una implementación diferente para esto.

7.2. PRUEBAS EN LA OBTENCIÓN DEL COEFICIENTE DE JACCARD

Para obtener el coeficiente de Jaccard en pares de páginas se necesitaron de tres procesos *Map/Reduce*. El primero, se encarga de calcular el número de usuarios que editaron una determinada página. El segundo, se encarga de agrupar cada página con su número total de usuarios. Finalmente, el tercer proceso es el que genera los pares y calcula el coeficiente de Jaccard. Los resultados obtenidos se muestran en la Tabla 7 y en la Figura 18.

Proceso MapReduce / N° páginas entrada	57.000	112.468	216.903	459.866	1.064.418
1° MapReduce	3,07	3,15	3,27	3,45	4,39
2° MapReduce	1,21	3,56	11,05	29,39	52,25
3° MapReduce	5,56	21,21	67,57	209,56	395,42
Tiempo Total	9,84	27,92	81,89	242,40	452,06

Tabla 7. Tiempos tomados para la obtención del coeficiente de Jaccard

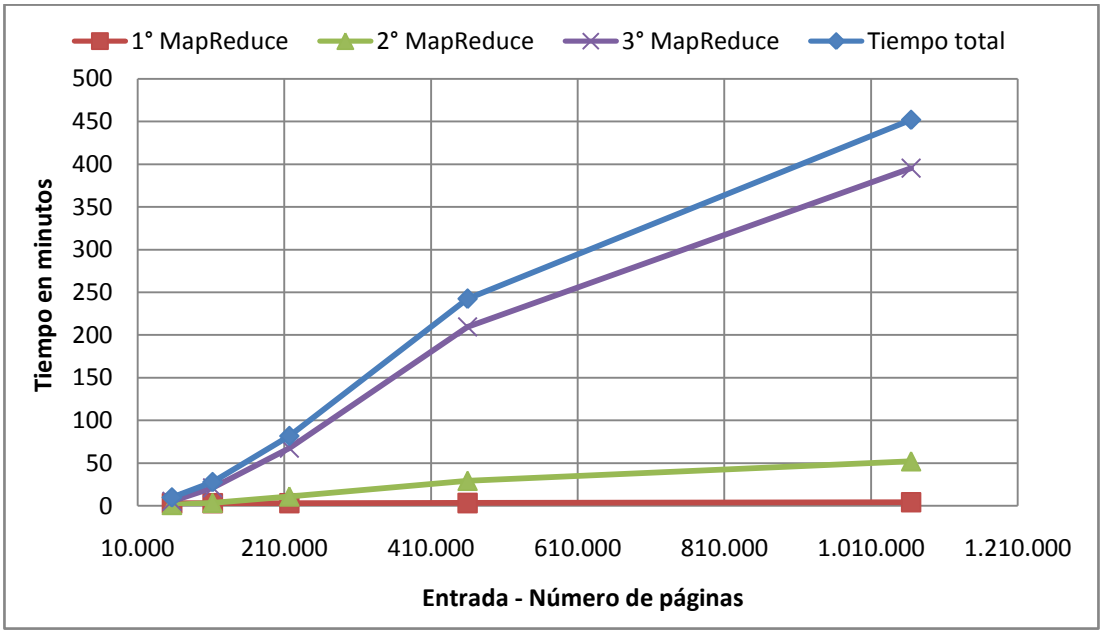

Figura 18. Resultado de pruebas para la obtención del coeficiente de Jaccard

Como se observa en el gráfico el primer proceso *Map/Reduce* no varía casi nada al incrementar el número de entrada. Sigue una tendencia casi constante, tanto así que de la primera a la última prueba solo tuvo un incremento de 1,32 minutos.

El segundo proceso si presenta una tendencia creciente evolutiva pero no muy significativa ya que entre la primera y última prueba hubo un incremento de menos de una hora. Esto es normal

ya que el segundo proceso tiene que iterar sobre una lista cuyo tamaño depende de los datos de entrada.

Otra observación importante es que a pequeñas cantidades de entradas del orden no mayor a las 200.000 páginas el primero y segundo proceso se ejecuta casi al mismo tiempo.

Como era de esperarse el tercer proceso Map reduce es el que más incide sobre el tiempo total; siendo así, que es el que marca la tendencia creciente casi exponencial para el tiempo total del proceso, presenta variaciones bruscas con respecto a la entrada.

Tal comportamiento era esperado, ya que durante este proceso los método map y reduce realizan operaciones que a pesar de que no son tan complejas se realizan al iterar sobre grandes conjuntos de datos, el map tiene que crear los pares de páginas y para esto recorre más de una vez sobre una lista, mientras que el reduce realiza el cálculo final para obtener el coeficiente de Jaccard. En resumen, se recorre varias veces una lista cuyo tamaño depende de los datos de entrada.

7.3. PRUEBAS EN SELECCIÓN DE LAS PÁGINAS A RECOMEDAR

En este proceso no se realiza ningún cálculo solo se encarga de ordenar la lista de páginas a recomendar dependiendo del valor de similitud que tenga cada página; esto se lo realiza en el reduce.

El tiempo depende del tamaño de los archivos de salida del proceso anterior (Obtención del coeficiente de Jaccard) ya que esta salida es la entrada para este proceso, los resultados obtenidos variando el tamaño de la entrada se muestran en la Tabla 8 y en la Figura 19.

# páginas	57.000	112.468	216.903	459.866	1.064.418
Tiempo en minutos	21,35	54,29	72,35	181,32	453,12

Tabla 8. Tiempos tomados para el proceso de selección de páginas a recomendar

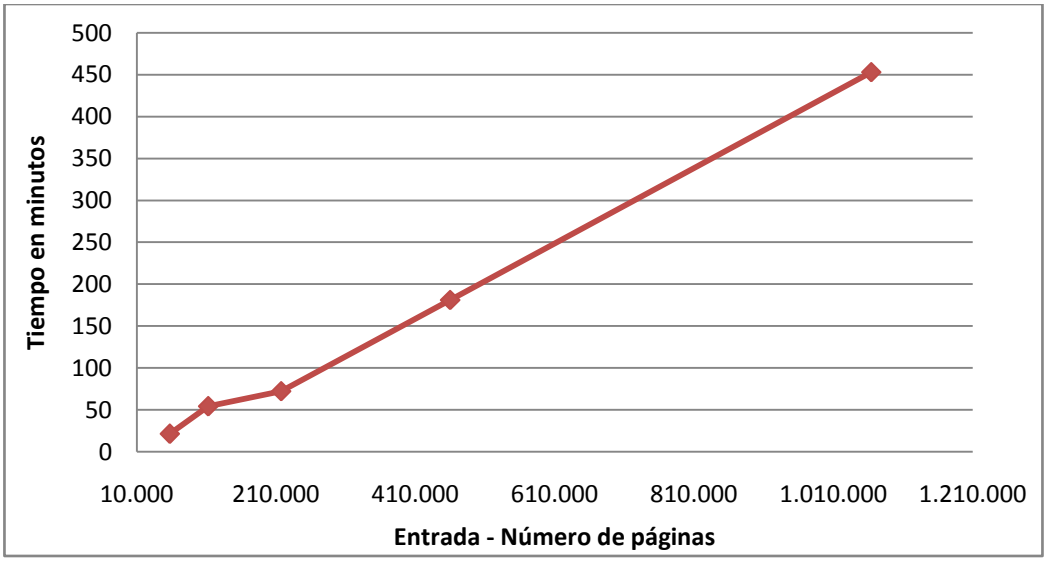

Figura 19. Resultado de pruebas para el proceso de selección de páginas a recomendar

Se observa algo muy peculiar con este proceso; al parecer los tiempos son muy parecidos a los que se obtuvieron en el proceso de filtrado. Esto nos hace pensar que el ordenamiento de la lista no presenta una demora tan significativa, y al igual que con el proceso de filtrado lo que toma más tiempo es emitir los map para cada par de página y el tiempo que se toma

Hadoop en ordenar y agrupar los map por clave.

Para ordenar la lista recibida en el reduce se utilizó el método Sort de la clase Collections de Java [24]. Este método utiliza un algoritmo de ordenamiento del orden O(n log n) [25] siendo este el mejor comportamiento que se puede tener para ordenar. Es por esto que la operación de ordenar la lista no influye mucho en el tiempo que demora el proceso, dando como resultado tiempos muy parecidos a los del proceso de filtrado ya que sin tomar en cuenta el ordenamiento de los datos, ambos procesos realizan operaciones muy parecidas.

7.4. RESULTADO FINAL

Algunos de los resultados finales obtenidos en lo que a contenido se refiere fueron los siguientes:

Búsqueda: Adolf Hitler

Wikis recomendadas: Segunda Guerra Mundial, Francia, Albert Einstein, Primera Guerra Mundial, Fascismo

Búsqueda: Ecuador

Wikis recomendadas: Brasil, Bolivia, Guatemala, Egipto, Colombia

Búsqueda: Windows Xp

Wikis recomendadas: Windows Vista, Virus informático, Internet Explorer, Windows 98,

HTML

Búsqueda: Copa Mundial de Futbol

Wikis recomendadas: Copa Mundial de Futbol 2006, Juegos Olímpicos, FIFA, Copa Mundial de Futbol 1962, Brasil

En las pruebas realizadas hasta el momento se observa que no se pierde la consistencia de datos entre el tema buscado y las wikis recomendadas, pero se ha hecho pruebas con temas donde de antemano se sabe que habrá gran cantidad de aportantes interesados en mejorar esas *wikis* que tratan de temas como personajes mundiales, países, eventos deportivos, tecnología, etc. Veamos que sucede al consultar temas variados, temas que probablemente no tengan usuarios dedicados por ejemplo música, juegos, etc.

Búsqueda: Ajedrez

Wikis recomendadas: Software, Algoritmo, Cáncer, Geografía, Química

Búsqueda: Luis Miguel

Wikis recomendadas: Paulina Rubio, Ensayo, Vicente Fox Quezada, Walt Disney, Louis Pasteur

Como se observa para el caso de Luis Miguel después de la primera recomendación la consistencia de datos se pierde, se aleja mucho en lo que al tema musical y artístico se refiere. Este resultado nos indica que gran cantidad de usuarios que aportaron a la *wiki* de Luis Miguel también aportaron sobre Vicente Fox, Walt Disney y Louis Pasteur, es decir no siguen una

misma temática de aportaciones, son usuarios que opinan sobre temas diversos. Por otro lado se observa que Paulina Rubio obtuvo el coeficiente de similitud de Jaccard más elevado ya que se encuentra primero, lo mismo pasa con las dos primeras recomendaciones de Ajedrez. En ambos casos las recomendaciones no se alejan mucho del tema buscado. Con esto comprobamos la utilidad del coeficiente de similitud de Jaccard como método para generar recomendaciones.

El resultado depende de los aportantes y recordemos que el coeficiente de similitud de Jaccard al ser un *valor porcentual* depende mucho del número de muestras que haya, es decir el número de usuarios que han aportado a una y ambas *wikis*.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. Utilizamos el coeficiente de similitud Jaccard como método para encontrar similitudes entre páginas. Este coeficiente utiliza teoría de conjuntos para hallar un porcentaje de similitud entre dos elementos de un conjunto. Logramos enfocar la teoría del Coeficiente de Jaccard para utilizarlo sobre los datos de la Wikipedia valiéndonos de los usuarios como elementos del conjunto página, implementamos este coeficiente en su versión MapReduce y de esta manera pudimos obtener todas las combinaciones posibles de páginas con su valor de similitud. Un proceso que no resulto muy costoso ya que para generar las recomendaciones se necesitan de tan solo $0,50 y para almacenar la salida de los procesos $1,50.

2. Observando los resultados obtenidos que se muestran en el apartado 7.4 del presente trabajo, podemos concluir que los usuarios de la Wikipedia no siempre pero si en algunas ocasiones aporta mayormente a *wikis* que tienen que ver con la misma temática y esto nos permite mantener una consistencia de información entre la *wiki* buscada y las recomendaciones pero no siempre será del 100% y en algunos casos puede que sea del 0%. Si lo que se quiere es tener recomendaciones basadas totalmente en contenido con un 100% de consistencia de datos, no es suficiente basar las recomendaciones en las aportaciones de los usuarios.

3. Para el caso particular de la Wikipedia que no almacena nada de información sobre gustos o preferencias de los usuarios, utilizar las aportaciones de los usuarios es un método valedero y rápido para generar recomendaciones y en algunos de los casos sin perder mucho la consistencia de datos.

4. En las pruebas realizadas se observó que el algoritmo obtiene su mayor eficiencia con un número de entradas del orden de las miles de páginas, al aumentar el tamaño a las millones de páginas los tiempos se disparan y se observa que se produce una escalabilidad lineal, estos tiempos podrían mejorarse aumentando el tamaño del clúster. Los procesos fueron desarrollados pensando en que se realizarían periódicamente con el fin de actualizar una base de datos, es por eso que consideramos que el mayor tiempo obtenido que fue de 15 horas sumando el proceso de obtención del coeficiente de similitud y el proceso de selección de páginas es un buen tiempo tomando en cuenta que se trabajó con 10 nodos para una entrada de más de un millón de páginas.

RECOMENDACIONES

1. Al realizar las salidas de los diferentes *Map/Reduce,* al principio utilizábamos los títulos de las páginas para identificarlas, estos eran concatenados y escritos en las diferentes salidas, comenzamos a notar un crecimiento desmesurado del tamaño de las salidas incluso para pequeñas entradas, llegando a tener archivos de hasta 1G para entradas de 1000 páginas. Esto retrasaba los procesos *Map/Reduce* que utilizaban esas salidas como entradas para sus operaciones. En base a esto, recomendamos evitar utilizar cadenas de caracteres tan largas para las salidas. En nuestro caso utilizamos los Id de las páginas que son enteros de longitud 10 y notamos una mejora en el tiempo de los procesos.

2. Los *bucles* internos dentro de los procesos *Map/Reduce* dependen mucho del tamaño de la entrada, conforme aumentábamos el tamaño de la entrada, estos *bucles* tomaban más de 10 minutos en terminar, durante este tiempo no hay actividad entre Hadoop y el proceso *Map/Reduce* ya que no se lee ni escribe datos en el *HDFS* pero si se realizan operaciones, Hadoop después de 10 minutos de detectar inactividad en un proceso procede a matar el proceso, es recomendable utilizar la instrucción reporter.progress() dentro de estos *bucles* para indicarle a Hadoop que el proceso sigue ahí.

3. Además de utilizar el índice de similitud de Jaccard, se podría mejorar el sistema de recomendación añadiéndole características como retroalimentación y *heurísticas* que permitan obtener recomendaciones basándose en otros criterios además de las aportaciones hechas por los usuarios.

GLOSARIO

AMAZON ELASTIC COMPUTE CLOUD (AMAZON EC2)

Servicio que permite a los clientes el alquiler de equipos en los que pueden ejecutar sus propias aplicaciones informáticas

AMAZON SIMPLE STORAGE SERVICE (AMAZON S3)

Es un servicio web en línea de almacenamiento ofrecido por Amazon Web Services.

API

Application Programming Interface, es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción

ARQUITECTURA CLIENTE/SERVIDOR

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta

BUCLES

Es una sentencia que se realiza repetidas veces a un trozo aislado de código, hasta que la condición asignada a dicho bucle deje de cumplirse

CARDINALIDAD

Determinación del número que corresponde a la cantidad de elementos de una colección.

CENTROS DE DATOS

Ubicación donde se concentran todos los recursos necesarios para el procesamiento de la información de una organización

CLOUD[9]

Librería Map/Reduce para Hadoop usada para enseñanza y apoyo en las investigaciones en el procesamiento de textos

CLÚSTER

Conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora

CÓDIGO FUENTE

Es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar un programa

DATANODES

Datanodes son aquellos que realizan de forma fiable y en condiciones de uso intenso o prolongado el trabajo en los sistemas de ficheros.

DATASET

Conjunto de información asociada a una fuente de datos

DUMPS

Nombre que se le da a los respaldos de los proyectos de la fundación Wikimedia

ECLIPSE

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido"

ESCALABILIDAD

Capacidad de un software o de un hardware de crecer, adaptándose a nuevos requisitos conforme cambian las necesidades del negocio

FRAMEWORK

Estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.

GFS

Google File System, es un sistema de almacenamiento basado en las necesidades de Google

GNOME

Es un entorno de escritorio e infraestructura de desarrollo para sistemas operativos Unix/GNU/Linux, compuesto enteramente de software libre

GZIP

Gzip es una abreviatura de GNU ZIP, un software libre GNU que reemplaza al programa

compress de UNIX

HDFS

Sistema de archivos distribuido usado por Hadoop

HEURÍSTICAS

Se denomina heurística a la capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines.

JOB

Representa un trabajo Map/Reduce en Hadoop

LA FUNDACIÓN WIKIMEDIA

Es la organización matriz de Wikipedia

LICENCIA GNU

Es una licencia creada por la Free Software Foundation, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

MAP/REDUCE

Es un Framework introducido por Google para soportar grandes conjuntos de datos sobre grupos de equipos con el apoyo de la computación distribuida

MÁQUINA VIRTUAL

Es un software que emula a un ordenador y puede ejecutar programas como si fuese un

ordenador real.

MYSQL

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario

NODO MAESTRO

Nodo desde el cual se controlan los procesos que se corren en el clúster

PARADIGMA

Conjunto de teorías acerca de una faceta de la realidad, que ofrece la posibilidad de resolver problemas a partir de sus principios fundamentales.

PLUGIN

Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica

PROGRAMACIÓN DISTRIBUIDA

Es un paradigma de programación enfocado en desarrollar sistemas distribuidos, abiertos, escalables, transparentes y tolerantes a fallos

PROGRAMAS ROBOT (BOTS)

Es un programa informático que realiza funciones muy diversas, imitando el comportamiento de un humano

PROMPT

Carácter o conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes

QUERY

Búsqueda en una base de datos

REPOSITORIO

Es un sitio centralizado donde se almacena y mantiene información digital

REPOSITORIO SUBVERSIÓN

Un sistema de revisiones de versiones de software

SISTEMA OPERATIVO FEDORA.

Es una distribución Linux para propósitos generales basada en RPM

SISTEMA OPERATIVO UBUNTU

Es una distribución de Linux basada en Debian GNU/Linux

SISTEMAS DE RECOMENDACIONES

Son un tipo específico de filtro de información, técnica que trata de presentar al usuarios ítems de información (películas, música, libros, noticias, páginas web) sobre las que el usuario está interesado

SSH

Secure Shell, protocolo informático que sirve para acceder a máquinas remotas

SVN

Subversión, un sistema de revisiones de versiones de software

TAG

Un tag es una palabra clave asociada a un artículo que hace referencia a él

TOLERANTE A FALLOS

Es la propiedad que permite a un sistema continuar operando adecuadamente en caso de una falla en alguno de sus componentes

TUPLAS

Es una secuencia ordenada de objetos, esto es, una lista con un número limitado de objetos

VALOR PORCENTUAL

Forma de expresar la concentración de cualquier solución.

WIKIPEDIAPAGE

Clase del paquete edu.umd.cloud9.collection.wikipedia que representa una página de la Wikipedia

WIKIS

Todo sitio web en donde colaboran múltiples autores.

REFERENCIAS BIBLIOGRÁFICAS

[1] Rodríguez-Salazar María Elena, Álvarez-Hernández Sergio, y Bravo-Núñez Ernesto, *Coeficientes de Asociación*, Plaza y Valdes, 2001 <http://books.google.com.ec/books?id=hitW9gbEGwoC> [Consultado: viernes 4 de septiembre del 2009].

[2] The Apache Software Foundation, "Apache Hadoop." <http://hadoop.apache.org/> [Consultado: viernes 4 de septiembre del 2009]

[3] Dean Jeffrey y Ghemawat Sanjay, "MapReduce: Simplified Data Processing on Large Clusters ," OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004..

[4] "Wikipedia - Wikipedia, la enciclopedia libre." <http://es.wikipedia.org/wiki/Wikipedia> [Consultado: miércoles 16 de septiembre del 2009]

[5] Ortega Felipe, "Wikipedia: A quantitative analysis.,". Resumen de Tesis Doctoral GSyC/Libresoft, Universidad Rey Juan Carlos, Madrid (Spain), Julio 2009.

[6] "Wikimedia Foundation." <http://wikimediafoundation.org> [Consultado: martes 8 de septiembre del 2009]

[7] "MediaWiki/es - MediaWiki." <http://www.mediawiki.org/wiki/MediaWiki/es> [Consultado: martes 8 de septiembre del 2009]

[8] White Tom, *Hadoop: The Definitive Guide*. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Cap. 2 Pág. 18 – Cap. 3 Pág. 42 – Cap. 4 Pág. 98.

[9] "Amazon Elastic MapReduce" <http://aws.amazon.com/elasticmapreduce/> [Consultado: lunes 21 de septiembre del 2009]

[10] "Amazon Elastic Compute Cloud (Amazon EC2)" <http://aws.amazon.com/ec2/> [Consultado: lunes 21 de septiembre del 2009]

[11] "Amazon Simple Storage Service (Amazon S3)" <http://aws.amazon.com/s3/> [Consultado: lunes 21 de septiembre del 2009]

[12] "Amazon Elastic MapReduce." <http://aws.amazon.com/elasticmapreduce/#pricing> [Consultado: martes 2 de septiembre del 2009]

[13] "Amazon Simple Storage Service (Amazon S3)." <http://aws.amazon.com/s3/#pricing> [Consultado: martes 2 de septiembre del 2009]

[14] "MWDumper - MediaWiki."<http://www.mediawiki.org/wiki/MWDumper> [Consultado: martes 8 de septiembre del 2009]

[15] "Cloud9: A Library for Hadoop." <http://www.umiacs.umd.edu/~jimmylin/cloud9/docs/index.html> [Consultado: lunes 3 de agosto del 2009]

[16] Borthakur Dhruba, "The Hadoop Distributed File System: Architecture and Design." <http://hadoop.apache.org/common/docs/r0.15.3/hdfs_design.pdf > [Consultado lunes 3 de agosto del 2009]

[17] "Hadoop Common Releases." <http://hadoop.apache.org/common/releases.html> [Consultado: lunes 3 de agosto del 2009]

[18] "Amazon Machine Image - Wikipedia, the free encyclopedia." <http://en.wikipedia.org/wiki/Amazon_Machine_Image> [Consultado: martes 15 de septiembre del 2009]

[19] "Usuario:AVBOT - Wikipedia, la enciclopedia libre." <http://es.wikipedia.org/wiki/Usuario:AVBOT> [Consultado: martes 15 de septiembre del 2009]

[20] Bank Jacob y Cole Benjamin, "Calculating the Jaccard Similarity Coefficient with Map Reduce for Entity Pairs in Wikipedia," December 16, 2008.

[21] "JDOM." <http://www.jdom.org/> [Consultado: martes 15 de septiembre del 2009]

[22] "S3Fox Organizer(S3Fox)." <http://www.s3fox.net/> [Consultado: lunes 3 de agosto del 2009]

[23] "Tabla hash - Wikipedia, la enciclopedia libre."<http://es.wikipedia.org/wiki/Tabla_hash> [Consultado: martes 15 de septiembre del 2009]

[24] "Collections (Java 2 Platform SE v1.4.2)." <http://java.sun.com/j2se/1.4.2/docs/api/java/util/Collections.html> [Consultado: martes 15 de septiembre del 2009]

[25] McGuire Tommy M., "O(n log n)." Junio 7, 2007 <http://www.crsr.net/Notes/BigO.html> [Consultado: martes 15 de septiembre del 2009]

[26] Wikimedia Commons, "Archivo: Wikipedia growth.png - Wikipedia, la enciclopedia libre," Archivo: Wikipedia growth.png,

<http://es.wikipedia.org/wiki/Archivo:Wikipedia_growth.png> [Consultado: Jueves 27 de agosto del 2009]

[27] “Sistemas de recomendaciones: herramientas para el filtrado de información en Internet”, <http://www.hipertext.net/web/pag227.htm> [Consultado: Lunes, 24 de Agosto del 2009]

[28] White Tom, *Hadoop: The Definitive Guide*. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Figure 2-3 Pág. 30.