ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
CCENTRO DE INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA

Búsquedas Avanzadas Tipo Grep de Tesis Realizadas en la ESPOL Utilizando el Paradigma Map-Reduce

Grace Aragundi [(1)]; Adriana Bedoya [(2)]; Cristina Abad [(3)]
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral
Campus Gustavo Galindo Velasco, Km. 30.5 vía Perimetral, Guayaquil, Ecuador
garagund@espol.edu.ec [(1)], abedoya@espol.edu.ec [(2)],cabad@fiec.espol.edu.ec [(3)]

Resumen

Actualmente la cantidad de información almacenada por las empresas, en sus bases de datos y a través de la Web, ha crecido de manera impresionante, esto ha originado la necesidad de implementar mecanismos alternos a los tradicionales, basados en el procesamiento en paralelo. En el presente documento se explica la implementación de un mecanismo de búsquedas avanzadas sobre las tesis y proyectos de graduación de la ESPOL, el cual es eficiente y escalable, basado en el paradigma MapReduce e implementado sobre el framework Hadoop, corriendo en clústeres levantados con el servicio EC2 de computación distribuida de los Amazon Web Services (AWS). Las pruebas realizadas utilizando diversas expresiones regulares nos permitieron crear consultas con mayores niveles de complejidad, logrando que los resultados reales generados a través de esta aplicación sean completamente representativos de lo que se esperaba encontrar.

Palabras Claves: *Hadoop, MapReduce, EC2, S3, Grep, Expresión Regular, Tesis, ESPOL.*

Abstract

Currently the amount of information stored by companies in their databases and through the Web, has grown dramatically, this has led to the need to implement alternative to traditional mechanisms, based on parallel processing. This document explains the implementation of an advanced search mechanism on the thesis and graduation projects in ESPOL, which is efficient and scalable, based on the Map Reduce paradigm and implemented on Hadoop framework, running on clusters raised with the distributed computing EC2 service of the Amazon Web Services (AWS). Tests were performed using different regular expressions, that enabled us to create queries with higher levels of complexity, making that the results generated through this application were fully representative of what is expected to find.

1. Introducción

En los últimos años el crecimiento de la cantidad de información que las empresas a nivel mundial están almacenando ha motivado que, día a día, más instituciones se encuentren adoptando nuevas metodologías que les permitan el acceso más ágil y el manejo más eficiente de sus datos.

El rendimiento es clave en cualquier proceso, y es aún más importante cuando la información que se maneja alcanza dimensiones en la escala de Gigabytes o Terabytes. El procesamiento de grandes volúmenes de datos podría tardarse incluso hasta varios días. Esto ha originado la necesidad de implementar mecanismos alternos a los tradicionales, basados en el procesamiento en paralelo.

Un ejemplo de un conjunto de datos de gran tamaño, es el conjunto de documentos de tesis y proyectos de graduación de los estudiantes de la ESPOL. En la actualidad, existe un sistema de búsquedas sobre títulos y palabras clave, mas no sobre los contenidos de las mismas. Si bien se puede implementar algún motor de búsqueda como Lucene para las búsquedas sobre los contenidos, estos sistemas están basados en índices invertidos y no permiten realizar búsquedas avanzadas basadas en expresiones regulares. Como parte del trabajo de investigación de los estudiantes y egresados, resultaría de beneficio poder realizar estas búsquedas. Herramientas como el grep de Unix permite realizar búsquedas basadas en expresiones regulares, pero utilizarlas para realizar búsquedas en las tesis de la ESPOL tomaría mucho tiempo debido a la gran cantidad de datos a procesar.

El presente trabajo implementa un mecanismo de búsquedas avanzadas sobre las tesis y proyectos de graduación de la ESPOL, el cual es eficiente y escalable, basado en el paradigma MapReduce e implementado sobre el framework Hadoop, corriendo en clústeres levantados con el servicio EC2 de computación distribuida de los Amazon Web Services (AWS).

2. Fundamentos Teóricos

2.1. Paradigma MapReduce

La enciclopedia libre Wikipedia define a MapReduce [1] como un framework introducido por Google, el cual soporta la computación en paralelo sobre grandes colecciones de datos en clústeres de computadoras. El framework ha sido inspirado en funciones map y reduce utilizadas comúnmente en los lenguajes funcionales, aunque con propósitos diferentes. Actualmente se han escrito implementaciones de MapReduce en C++, Java, Python y otros lenguajes.

MapReduce divide el procesamiento en dos fases: Map y Reduce. La función map() posee la característica de trabajar sobre grandes volúmenes de datos. Estos datos son divididos en dos o más partes. Cada una de estas partes contiene colecciones de registros o líneas de texto.

La función reduce() se ejecuta para cada elemento de cada lista de valores intermedios que recibe. El resultado final se obtiene mediante la recopilación e interpretación de los resultados de todos los procesos que se ejecutaron.

2.2. Expresiones Regulares

Las expresiones regulares están constituidas por una serie de caracteres que forman un patrón, normalmente representativo de otro grupo de caracteres mayor. Es decir, en una expresión regular se describe un conjunto de cadenas sin enumerar sus elementos, de tal forma que podemos comparar el patrón con otro conjunto de caracteres para ver las coincidencias.

En la Tabla 1, se muestra la sintaxis más utilizada en la formación de expresiones regulares.

Tabla 1. Ejemplo de sintaxis utilizada.

Carácter	Significado	Ejemplo
Punto "."	Cualquier otro carácter excepto el salto de línea.	a.b
Corchetes "[ ]"	Agrupa caracteres de acuerdo a una clase o tipo.	[a-z]
Barra "|"	Separa alternativas u opciones.	a|b|c
Signo de dólar "$"	Representa el final de la cadena de caracteres o el final de la línea.	\.$
Acento circunflejo "^"	Representa el inicio de la cadena.	^[a-z]
	Permite especificar caracteres que queremos que no estén en la cadena.	[^\w]
Signo de interrogación "?"	Sirve para especificar que una parte de la búsqueda es opcional.	ob?scuridad
Asterisco "*"	Sirve para encontrar algo que se encuentra repetido 0 o más veces.	la*
Signo de suma "+"	Se utiliza para encontrar una cadena que se encuentra repetida 1 o más veces.	a+
Paréntesis "()"	Sirven para agrupar caracteres.	al (este|oeste|norte|sur) de
Llaves "{}"	Las llaves permiten definir valores mínimos y máximos en una expresión regular.	\w{2,4}
Contrabarra o barra invertida "\"	La barra invertida no se utiliza nunca por sí sola, sino en combinación con otros caracteres.	\d Dígito del 0 al 9 \D Cualquier carácter que no sea un dígito del 0 al 9 \w Cualquier carácter alfanumérico \W Cualquier carácter no alfanumérico \s Espacio en blanco

2.2. Grep

Grep es una utilidad de línea de comandos escrita originalmente para ser usada con el sistema operativo Unix. Usualmente, grep toma una expresión regular de la línea de comandos, lee la entrada estándar o una

lista de archivos, e imprime las líneas que contengan coincidencias para la expresión regular. [2]

```
$ grep [opciones] [expresión regular] [archivo]
```

3. Desarrollo de la Solución

3.1. Diseño

El mecanismo planteado en el presente trabajo se basa en dos fases: una etapa de pre-procesamiento de los documentos digitalizados de las tesis y el proceso que realiza la búsqueda tipo Grep.

El sistema de búsquedas tipo grep es alimentado por un conjunto de tesis convertidas a formato de texto plano, sobre las cuales se realizan las consultas solicitadas por los usuarios a través de expresiones regulares.

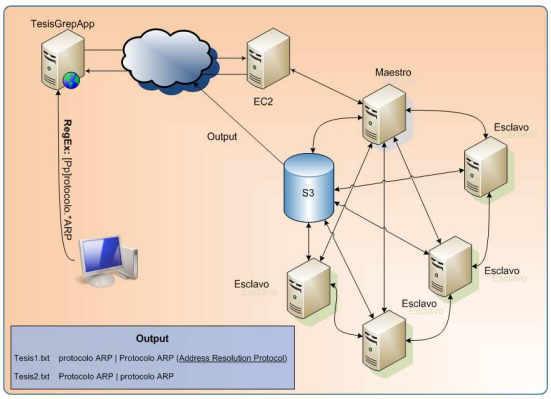

Figura 2. Arquitectura general del sistema.

3.2. Implementación

3.2.1. Pre-procesamiento de datos.

Inicialmente los archivos digitalizados de las tesis se encontraban en formato PDF [1], por lo cual surgió la necesidad de someterlos a una etapa de pre-procesamiento con la finalidad de realizar la conversión de los mismos a texto plano (*.txt). Para la conversión se utilizó un proceso MapReduce sencillo como se indica en la Figura 3.

[1] El Portable Document Format (PDF) es un formato de almacenamiento de documentos soportado por los sistemas operativos Windows, Unix/Linux y Mac. Systems.

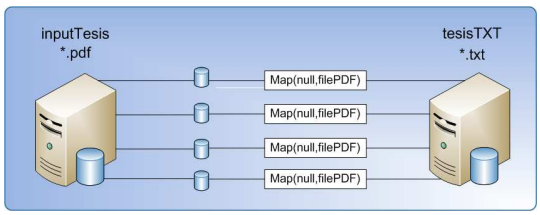

Figura 3. Proceso MapReduce para la conversión de archivos de entrada.

En la fase Map, luego de recibir cada archivo se utilizó la librería PDFBox [3] de Apache, por medio de la cual se extrajo el texto de cada fichero y se generó un archivo txt con el nombre correspondiente a cada documento de tesis analizado. Este archivo no contiene saltos de línea, por lo tanto su contenido completo se encuentra en una simple línea de gran tamaño, y puede ser procesado posteriormente por un mapper utilizando TextInputFormat.

3.2.2. Proceso para la búsqueda tipo Grep.

El proceso MapReduce inicia con una consulta escrita por el usuario utilizando expresiones regulares, busca las coincidencias y retorna un archivo formado por pares <nombreArchivo, ocurrencia1 | ocurrencia2 | …>. El diseño general se muestra en la Figura 4.

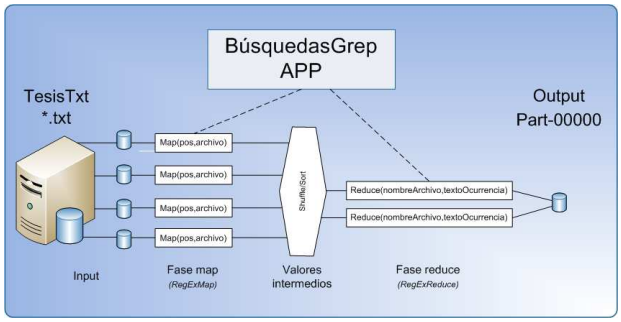

Figura 4. Proceso MapReduce para la búsqueda tipo Grep.

Internamente la función map toma el nombre del archivo que se está analizando en ese momento y realiza la búsqueda del texto solicitado por el usuario en el contenido de la línea. En la función reduce se concatenan todas las ocurrencias encontradas correspondientes al mismo archivo.

4. Resultados

Al momento de ejecutar la aplicación de manera distribuida [4], se usaron dos Servicios Web de

Amazon (AWS), a los cuales pudimos acceder de manera gratuita por medio del programa de fondos AWS in Education [5].

Las tesis de la ESPOL se colocaron en la nube con ayuda del servicio conocido como S3, el cual permite almacenamiento de información de manera escalable [6] y el clúster de tamaño variable que ofrece la capacidad de procesamiento para el sistema es provisto por el servicio EC2 [7].

Para efectuar las pruebas se varió la cantidad de nodos levantados en el clúster de EC2 con la finalidad de evaluar el rendimiento del sistema.

Los tiempos de respuesta se midieron al realizar consultas con 10 nodos como valor mínimo y se incrementó hasta 20 nodos.

Se tomaron cinco mediciones de tiempo de respuesta para cinco tamaños distintos del clúster de EC2.

En la Figura 5 podemos comprobar que a medida que incrementamos el número de nodos levantados, el tiempo de respuesta disminuye proporcionalmente, excepto con las pruebas realizadas sobre 20 nodos en donde los tiempos de respuesta son muy parecidos a los tiempos tomados con 18 nodos.

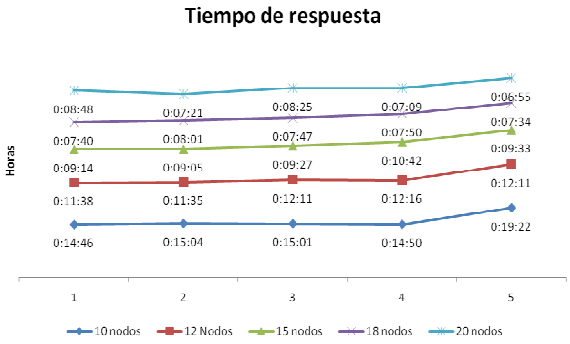

Figura 5. Tiempos de respuesta.

Los tiempos promedio de respuesta para los distintos tamaños del clúster se detallan en la Tabla 2.

Tabla 2. Tiempo promedio de ejecución.

# Nodos	Tiempo Prom. (hh:mm:ss)
10	0:15:49
12	0:11:58
15	0:09:36
18	0:07:46
20	0:07:44

5. Conclusiones

Los motores de búsqueda comúnmente hacen uso de índices invertidos previamente elaborados, por medio de los cuales se obtienen rápidamente los resultados para las consultas realizadas por los usuarios. Esta es la solución más óptima cuando se trata de consultas simples, sin embargo este mecanismo no soporta la realización de búsquedas más específicas dentro del contenido de un amplio conjunto de datos. El Sistema de Búsquedas que hemos detallado a lo largo de este documento trabaja con expresiones regulares, las cuales permiten crear consultas con mayores niveles de complejidad, logrando que los resultados generados por la aplicación sean 100% representativos de lo que esperábamos encontrar.

El uso de nuevas herramientas gratuitas de procesamiento masivo y distribuido de datos como Hadoop, permiten a las empresas acceder a la posibilidad de escalar de manera veloz en función de sus requerimientos, sin tener que depender de la capacidad de procesamiento de un solo equipo de trabajo o de cantidades pequeñas de información sobre las cuales realizar análisis.

6. Recomendaciones

Implementar una interfaz más amigable que permita a usuarios que no sean expertos en la sintaxis de expresiones regulares, elaborar consultas a través de pautas y criterios generales, permitiéndoles construir sus propias expresiones de forma abstracta pero obteniendo resultados igual de eficientes.

El ordenamiento de los resultados de cada consulta podría basarse en mejores criterios, presentando como primer resultado a aquel en el cual se haya detectado más coincidencias para la expresión buscada y en último lugar se mostraría el detalle del archivo con el menor número de ocurrencias. Nuestra propuesta muestra los resultados en orden alfabético, sin tomar en consideración los mejores resultados para el usuario.

7. Referencias

[1] Wikipedia. [En línea] [Citado el: 3 de Octubre de 2009.]. http://en.wikipedia.org/wiki/MapReduce.

[2] Wikipedia. [En línea] [Citado el: 3 de Octubre de 2009.] http://es.wikipedia.org/wiki/Expresión_regular.

[3] The Apache Software Foundation. [En línea] [Citado el: 3 de Octubre de 2009.] http://incubator.apache.org/pdfbox/download.html#pdfbox.
[4] Wikipedia. [En línea] [Citado el: 3 de Octubre de 2009.] http://en.wikipedia.org/wiki/Cloud_computing.
[5] Amazon. [En línea] [Citado el: 3 de Octubre de 2009.] http://aws.amazon.com/education/.
[6] Amazon. [En línea] [Citado el: 3 de Octubre de 2009.] http://aws.amazon.com/s3.
[7] Amazon. [En línea] [Citado el: 3 de Octubre de 2009.] http://aws.amazon.com/ec2.