ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

INFORME DE MATERIA DE GRADUACIÓN

"BÚSQUEDAS AVANZADAS TIPO GREP DE TESIS REALIZADAS EN LA ESPOL UTILIZANDO EL PARADIGMA MAP-REDUCE"

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN
ESPECIALIZACIÓN SISTEMAS DE INFORMACIÓN

INGENIERO EN COMPUTACIÓN
ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

Presentada por:

GRACE VERÓNICA ARAGUNDI RIVAS
ADRIANA DEL ROCÍO BEDOYA ZAMBRANO

Guayaquil - Ecuador

2009

AGRADECIMIENTO

A Dios,
por habernos dado la fortaleza necesaria para superar los obstáculos.

A nuestros padres,
por brindarnos un hogar cálido y enseñarnos que la perseverancia y el esfuerzo son cruciales para el éxito.

A la Ingeniera Cristina Abad,
por sus valiosas enseñanzas y por estimularnos para seguir creciendo intelectualmente.

A nuestros amigos,
que no dudaron en ayudarnos cuando más los necesitamos.

DEDICATORIA

Con mucho cariño,
a nuestras familias y amigos,
por su apoyo incondicional.

TRIBUNAL DE GRADO

DIRECTORA DEL PROYECTO DE GRADUACIÓN

MsC. Cristina Abad Robalino

PROFESOR DELEGADO POR EL DECANO

Ing. Fabricio Echeverría Briones

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto 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).

Grace Verónica Aragundi Rivas

Adriana del Rocío Bedoya Zambrano

RESUMEN

En el presente trabajo de graduación se detalla la implementación de un Sistema de Búsquedas tipo Grep, es decir, basado en expresiones regulares que permite a los usuarios realizar consultas sobre el contenido de las tesis digitalizadas en la ESPOL utilizando para esto la programación en paralelo.

En el capítulo 1 se explica el funcionamiento del Paradigma MapReduce, las fases que lo componen y las implementaciones existentes. Entre ellas se encuentra Hadoop, de código abierto, el cual constituye la base para nuestro proyecto.

En el capítulo 2 se indican los conceptos relacionados con las expresiones regulares, se muestran ejemplos sobre la sintaxis y los operadores utilizados. Además, se utiliza la clasificación de los motores de búsqueda de acuerdo al uso de las expresiones regulares con la finalidad de introducir los conceptos del funcionamiento del comando Grep de Unix/Linux.

El capítulo 3 inicia con el análisis del código de un Grep Distribuido implementado por Hadoop y contenido en su librería de ejemplos, el cual sirvió de referencia para nuestro proyecto. Adicionalmente se encuentran el objetivo, la motivación y el alcance del sistema de búsquedas propuesto.

También se detalla su diseño, se expone cómo se preparan los datos en una etapa de pre-procesamiento y se explica el funcionamiento de las fases MapReduce empleadas.

Dentro del capítulo 4 encontramos información sobre las herramientas utilizadas para la implementación. A más de esto se mencionan las pruebas de rendimientos realizadas sobre la aplicación y un análisis de los resultados obtenidos.

Las conclusiones y recomendaciones sobre el sistema se mencionan al final de este trabajo de graduación.

ÍNDICE GENERAL

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

INTRODUCCIÓN

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. Por tal motivo, día a día, más instituciones se encuentran 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. Tal es el caso del paradigma MapReduce, a través del proyecto Hadoop de Apache de código abierto.

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).

CAPÍTULO I

1 PARADIGMA MAPREDUCE

1.1. Introducción

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.

1.2. Fases MapReduce

MapReduce divide el procesamiento en dos fases: Map y Reduce. Cada una de estas fases utiliza pares <clave-valor> como entradas y salidas.

Map<k1,v1> -> list<k2,v2>

Reduce<k2, list (v2)> -> list(v2)

1.2.1. Función map

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.

Una función map() es ejecutada para cada porción de datos por separado, con la finalidad de calcular un conjunto de valores intermedios basados en el procesamiento de cada registro. MapReduce agrupa los valores de acuerdo a la clave intermedia y posteriormente los envía a la función reduce().

1.2.2. Función reduce

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.

1.2.3. Forma general del proceso MapReduce

```
map(function, list) {
    foreach element in list {
        value = function(element)
        intermediateResult.add(value)
    }
}
```

```
reduce(function, list, init) {
    result = init
    foreach value in list {
        result = function(result, value)
    }
  outputResult.add(result)
}
```

1.3. Aplicaciones

MapReduce actualmente es ampliamente utilizado. Las aplicaciones principales están relacionadas con:

- Grep distribuido
- Ordenamiento distribuido
- Construcción de índices invertidos
- Sistemas de recomendación
- Análisis de logs
- Problemas de aprendizaje de máquina (machine learning)

1.4. Implementaciones existentes

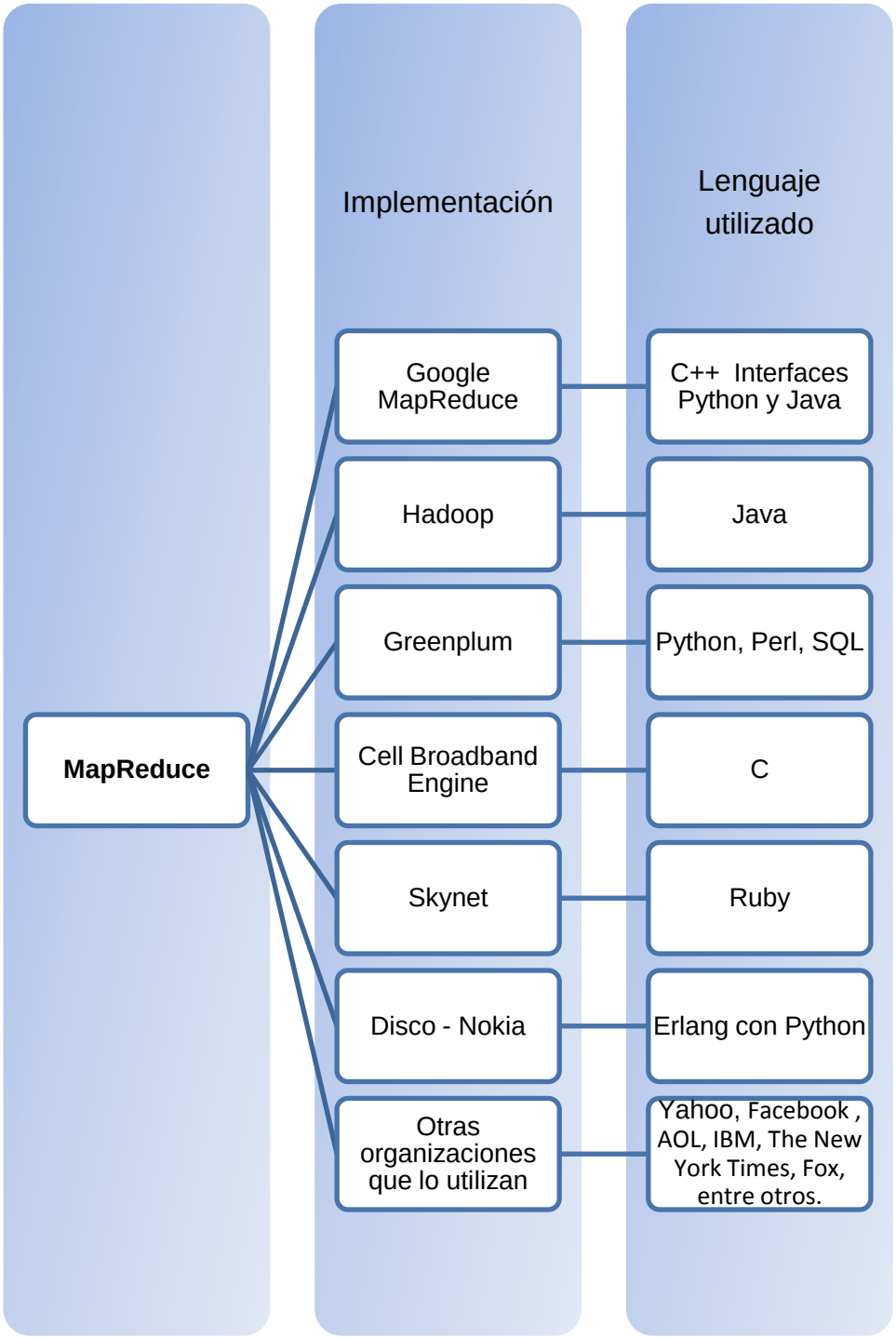


Gráfico 1: Implementaciones existentes de MapReduce

CAPÍTULO II

2 EXPRESIONES REGULARES

2.1. Definición

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.

2.2. Descripción de la sintaxis utilizada

- **Punto "."**

  El punto es interpretado como cualquier otro carácter excepto el salto de línea.

  RegEx: 'a.b'

  Coincidencias: aab…azb a0b…a9b a@b a#b…aáb…aúb

- **Corchetes "[ ]"**

  Los corchetes se utilizan para agrupar caracteres de acuerdo a una clase o tipo. Dentro de los corchetes es posible utilizar el guión "-" para especificar rangos de caracteres.

RegEx: ‘[a-z]’

Equivalencia: Rango para las letras minúsculas de la a hasta la z

- **Barra "|"**

  La barra vertical se utiliza para separar alternativas u opciones. Si por lo menos una de ellas coincide, el patrón será cierto.

  RegEx: ‘ a|b|c’

  Coincidencias: a b c

- **Signo de dólar "$"**

  Representa el final de la cadena de caracteres o el final de la línea, si se utiliza el modo multi-línea. No representa un carácter en especial sino una posición.

  RegEx: ‘\.$’

  Coincidencias: todos los lugares donde un punto finalice la línea

- **Acento circunflejo "^"**

  Este caracter "^" tiene doble funcionalidad:

  Representa el inicio de la cadena.

  RegEx: ‘^[a-z]’

  Coincidencias: todos los párrafos que den inicio con una letra minúscula

Permite especificar caracteres que queremos que no estén en la cadena.

RegEx: ‘ [^\w]’

Coincidencias: cualquier carácter que no sea alfanumérico o un espacio

- **Signo de interrogación "?"**

El signo de pregunta tiene varias funciones dentro del lenguaje de las expresiones regulares.

Sirve para especificar que una parte de la búsqueda es opcional.

RegEx: ‘ob?scuridad’

Coincidencias: oscuridad obscuridad

RegEx: ‘Nov(?:\.|iembre|ember)?’

Coincidencias: Nov Nov. Noviembre November

Los paréntesis definen grupos "anónimos", sin embargo el signo de pregunta en conjunto con los paréntesis triangulares "<>" permiten nombrar estos grupos.

RegEx: ‘^(?<Día>\d\d)/(?<Mes>\d\d)/(?<Año>\d\d\d\d)$’

Coincidencias: los primeros dos dígitos llevarán la etiqueta "Día", los segundos la etiqueta "Mes" y los últimos cuatro dígitos llevarán la etiqueta "Año".

- **Asterisco "*"**

  El asterisco sirve para encontrar algo que se encuentra repetido 0 o más veces.

  RegEx: ‘1a*’

  Coincidencias: 1 1a 1aa 1aaa 1aaaa

- **Signo de suma "+"**

  Se utiliza para encontrar una cadena que se encuentra repetida 1 o más veces.

  RegEx: ‘a+’

  Coincidencias: a aa aaa aaaa aaaaa

- **Paréntesis "()"**

  Los paréntesis sirven para agrupar caracteres. Estos grupos establecen una "etiqueta" o "punto de referencia".

  RegEx: ‘al (este|oeste|norte|sur) de’

  Coincidencias: textos que den indicaciones por medio de puntos cardinales

- **Llaves "{}"**

  Las llaves permiten definir valores mínimos y máximos en una expresión regular.

RegEx: ‘\w{2,4}’

Coincidencias: palabras de dos a cuatro letras

El valor máximo puede ser omitido.

RegEx: ‘\w{3,}’

Coincidencias: palabras de tres letras o más

RegEx: ‘\w{8}’

Coincidencias: palabras de ocho letras exactamente

Se utilizan junto a un carácter o patrón para especificar el número de veces que se desea que se repita.

RegEx: ‘^\d{2}/\d{2}/\d{4}$’

Equivalencia: ^\d\d/\d\d/\d\d\d\d$

Valida un formato de fecha

- **Contrabarra o barra invertida "\"**

La barra invertida no se utiliza nunca por sí sola, sino en combinación con otros caracteres, para conseguir un significado diferente que ayude a escribir mejores expresiones regulares. A continuación se muestra una lista detallada.

Signo	Representación
\t	Tabulador
\r	Retorno de carro o regreso al inicio
\n	Salto de línea
\a	Sonido que se produce al imprimir este carácter.
\e	Tecla "Esc"

\f	Salto de página
\v	Tabulador vertical
\x	Caracteres ASCII o ANSI. Ejemplo "\xA9" para símbolo de derechos de autor
\u	Caracteres Unicode. Ejemplo "\u00A2" para símbolo de centavos
\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
\S	Cualquier carácter que no sea un espacio en blanco
\A	Inicio de la cadena (posición)
\Z	Final de la cadena (posición)
\b	Marca el inicio y el final de una palabra
\B	Marca la posición entre dos caracteres alfanuméricos o dos no-alfanuméricos

Tabla 1: Usos de la barra invertida “\” en expresiones regulares

2.3. Las expresiones regulares y los motores de búsqueda

Para poder utilizar las expresiones regulares al programar es necesario tener acceso a un motor de búsqueda con la capacidad de utilizarlas. Es posible clasificar los motores disponibles en dos tipos: Motores para el programador y Motores para el usuario final. (2)

2.3.1. Motores de búsqueda para el programador

Permiten automatizar el proceso de búsqueda de modo que sea posible utilizarlo muchas veces para un propósito específico.

A continuación se detallan algunas de las herramientas de programación disponibles que ofrecen motores de búsqueda con soporte a expresiones regulares.

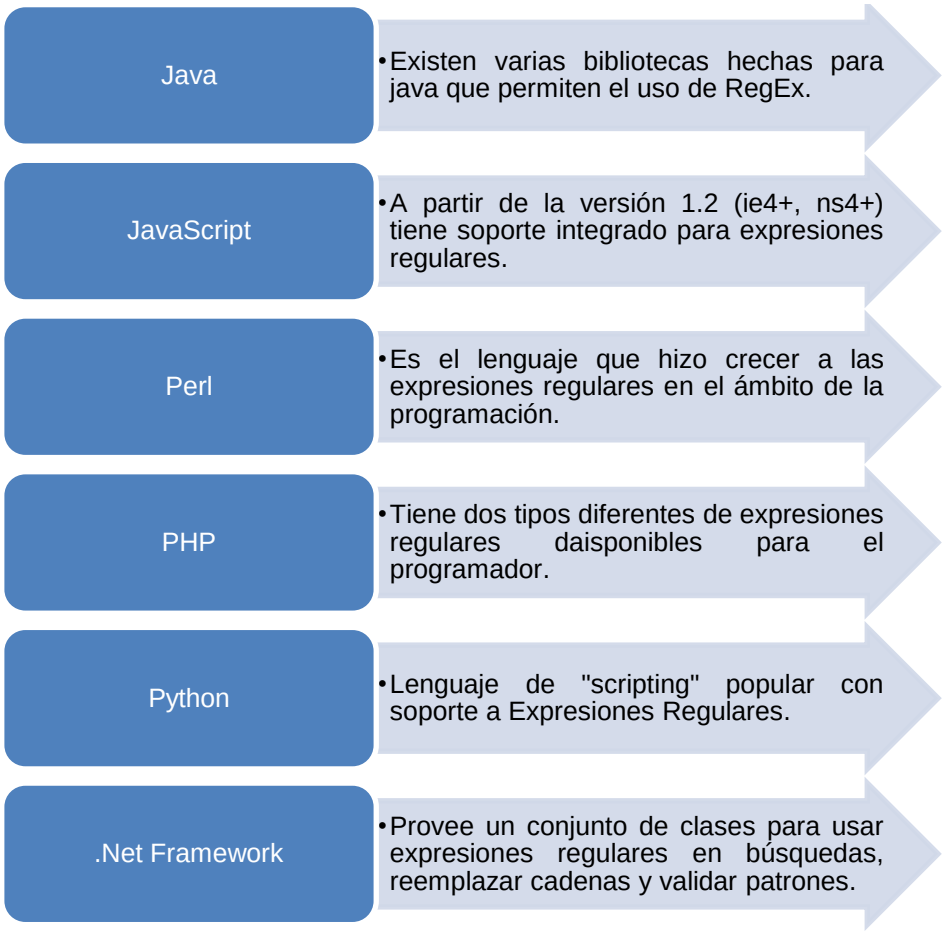


Gráfico 2: Herramientas de programación que soportan expresiones regulares

2.3.2. Motores de búsqueda para el usuario final

Son programas que permiten realizar búsquedas avanzadas sobre el contenido de un archivo o sobre un texto extraído y colocado en el programa.

Los principales programas se mencionan a continuación:

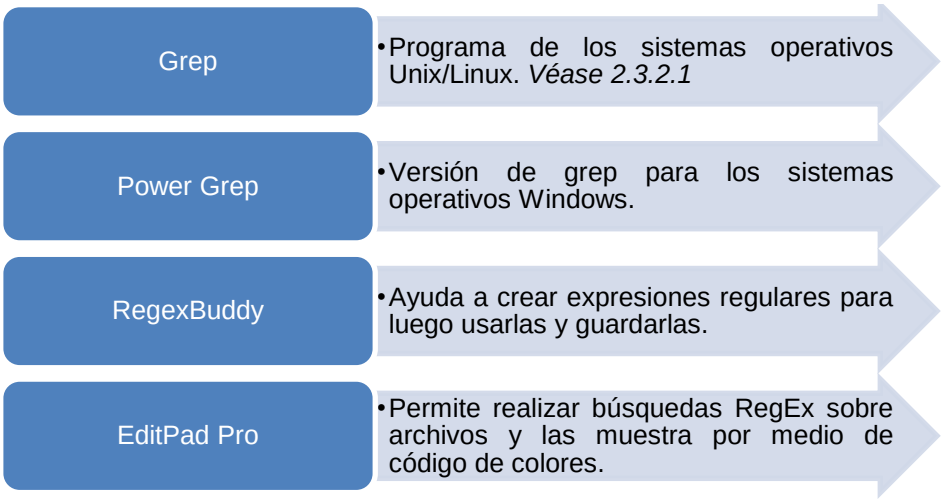


Gráfico 3: Programas que permiten realizar búsquedas con expresiones regulares

2.3.2.1. 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. (3)

Su estructura es:

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

CAPÍTULO III

3 BÚSQUEDAS TIPO GREP UTILIZANDO MAPREDUCE

3.1. Antecedentes: Grep distribuido de Hadoop

Dentro de la librería de ejemplos de Hadoop podemos encontrar la implementación de un ***Grep Distribuido.*** Este ejemplo extrae todas las cadenas de caracteres de los archivos de texto del directorio de entrada que concuerden con la expresión regular especificada como parámetro, además cuenta el número de veces que han sido encontradas.

3.1.1. Diseño del grep distribuido

Este programa ejecuta dos trabajos MapReduce en secuencia. El primer trabajo se encarga de contar el número de ocurrencias; en cambio, el segundo trabajo ordena las ocurrencias de acuerdo a la frecuencia.

Cada función map del primer trabajo toma una línea como entrada y busca las concordancias con la expresión regular del usuario. Extrae las cadenas halladas y emite pares (cadena,1). Cada función reduce se encarga de sumar las frecuencias de cada cadena.

En la segunda fase, la salida del primer trabajo se toma como entrada y se procede al ordenamiento. Para esta etapa se utiliza un solo reduce, por lo que se obtiene solo un archivo de salida, el cual muestra los datos ordenados de forma descendiente. Cada línea del archivo contiene el total del conteo y la cadena relacionada.

3.1.2. Ejecución del grep distribuido

El comando para ejecutar el ejemplo del grep distribuido se basa en el siguiente formato:

```
$ hadoop jar $HADOOP_HOME/hadoop-0.18.3-patched-examples.jar grep
<indir> <outdir> <regex> [<group>]
```

En donde:

<indir> es el directorio de entrada de los datos.

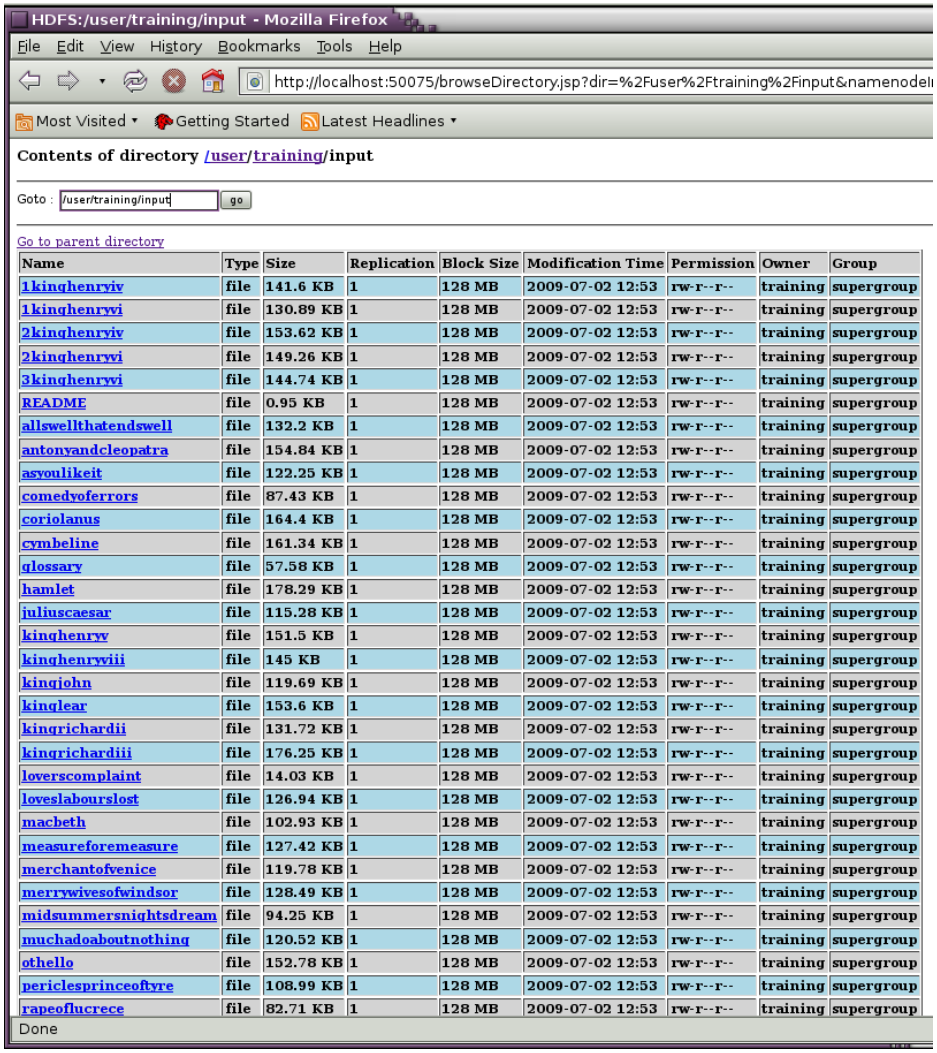
<outdir> es el directorio de salida del grep.

<regex> es la expresión regular que incluye la cadena de caracteres que se buscará dentro del directorio <indir>.

Para mostrar la funcionalidad de este ejemplo, vamos a utilizar la siguiente línea desde la terminal:

```
$ hadoop jar $HADOOP_HOME/hadoop-0.18.3-patched-examples.jar grep
input  grep_output "[Ww]herefore"
```

En *input* se encuentran almacenadas las obras de Shakespeare que nos sirvieron como archivos de entrada en nuestro ejemplo.



Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
1kinghenryiv	file	141.6 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
1kinghenryvi	file	130.89 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
2kinghenryiv	file	153.62 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
2kinghenryvi	file	149.26 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
3kinghenryvi	file	144.74 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
README	file	0.95 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
allswellthatendswell	file	132.2 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
antonyandcleopatra	file	154.84 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
asyoulikeit	file	122.25 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
comedyoferrors	file	87.43 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
coriolanus	file	164.4 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
cymbeline	file	161.34 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
glossary	file	57.58 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
hamlet	file	178.29 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
juliuscaesar	file	115.28 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
kinghenryv	file	151.5 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
kinghenryviii	file	145 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
kingjohn	file	119.69 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
kinglear	file	153.6 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
kingrichardii	file	131.72 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
kingrichardiii	file	176.25 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
loverscomplaint	file	14.03 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
loveslabourslost	file	126.94 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
macbeth	file	102.93 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
measureforemeasure	file	127.42 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
merchantofvenice	file	119.78 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
merrywivesofwindsor	file	128.49 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
midsummersnightsdream	file	94.25 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
muchadoaboutnothing	file	120.52 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
othello	file	152.78 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
periclesprinceoftyre	file	108.99 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup
rapeoflucrece	file	82.71 KB	1	128 MB	2009-07-02 12:53	rw-r--r--	training	supergroup

Gráfico 4: Contenido del directorio de entrada *input* del grep distribuido

En *grep_output* se almacenó el archivo de salida del grep luego de la ejecución.

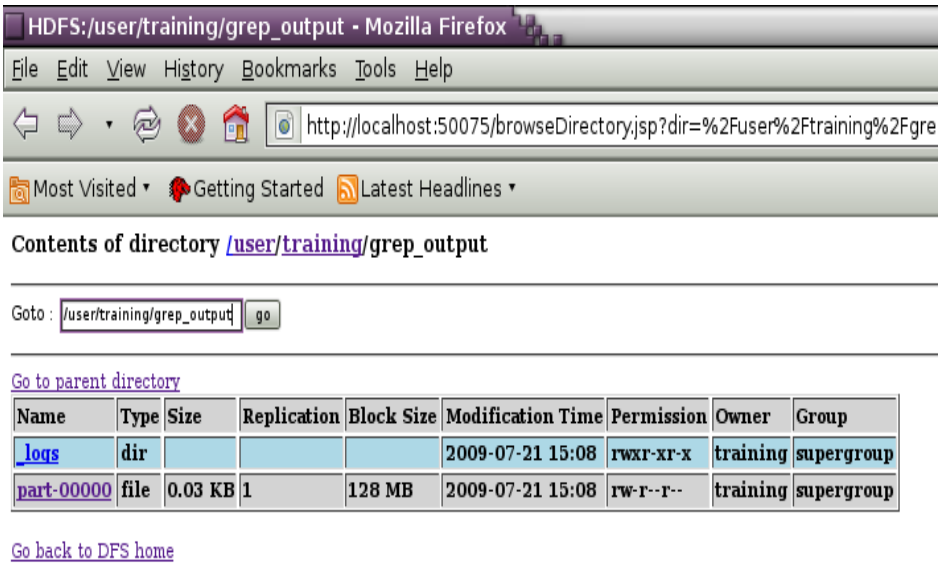


Gráfico 5: Contenido del directorio de salida *grep_output* del Grep Distribuido

La expresión regular que se utilizó es "[Ww]herefore", que nos permitió buscar la palabra wherefore en mayúsculas y minúsculas dentro del directorio de entrada.

Luego de la ejecución del proceso MapReduce del grep distribuido se creará el archivo de salida part-00000. *(Véanse Gráfico 6 y Gráfico 7)*

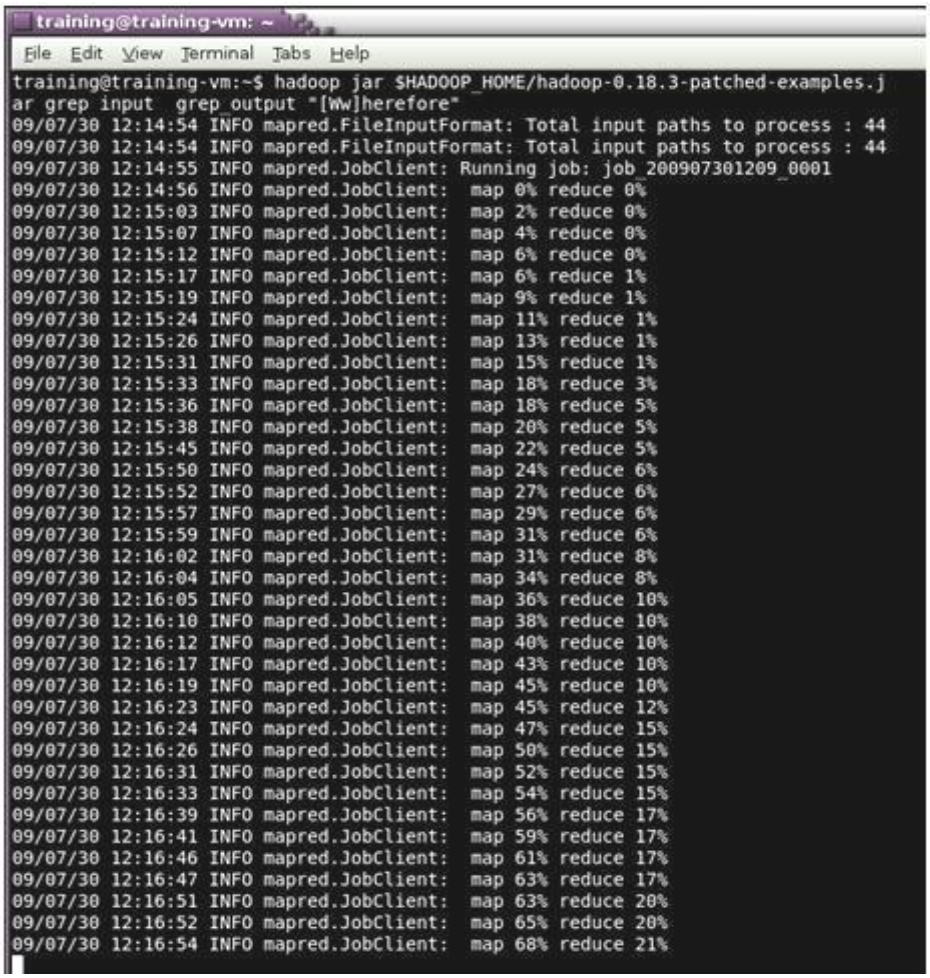


Gráfico 6: Ejecución del grep distribuido

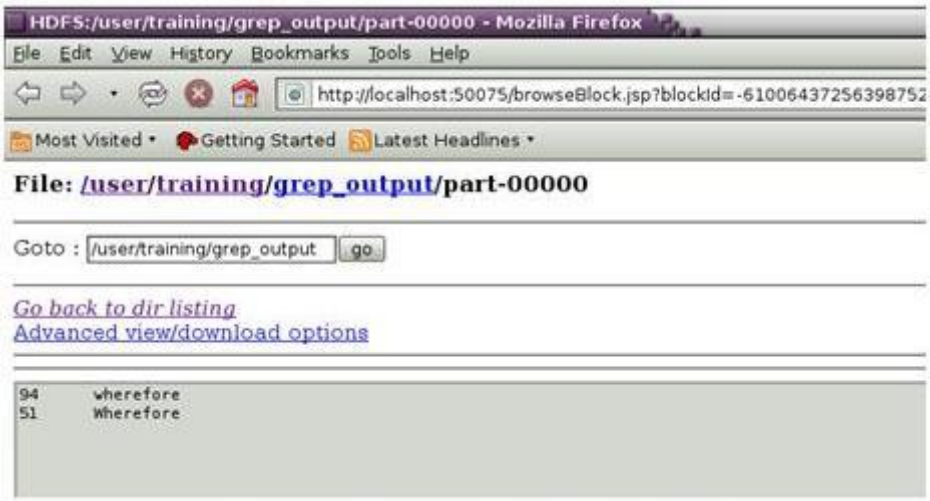


Gráfico 7: Archivo de salida del grep distribuido

3.2. Objetivo

Diseñar y evaluar un módulo de búsquedas avanzadas que permita realizar búsquedas basadas en expresiones regulares sobre el conjunto de tesis y proyectos de graduación de la ESPOL.

3.3. Motivación

Actualmente en la ESPOL existen alrededor de 13 GBs en documentos de tesis, digitalizados y almacenados en el Centro de Información Bibliotecario (CIB). La Biblioteca Virtual cuenta con una opción de búsquedas de tesis que permite obtener información a partir de ciertos criterios como título, autor, unidad académica, año y descriptor.

Los estudiantes al momento de seleccionar un tema de tesis suelen perder mucho tiempo en caso de considerarse que ya existen anteriores tesis relacionadas con la que están proponiendo. Para comprobar esto, utilizan el sistema de búsquedas existente, sin embargo, como dicho sistema no acepta consultas sobre el contenido, los alumnos deben (en ocasiones) verificar una a una las tesis publicadas de la facultad.

Esto nos lleva a la necesidad de implementar un sistema de búsquedas mucho más avanzado, a partir de expresiones regulares que realicen la consulta no sólo en ciertas características del documento sino en todo su contenido, ayudando a los estudiantes a reducir tiempos al momento de realizar sus consultas.

3.4. Diseño

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. Por medio del gráfico 8 se puede visualizar su arquitectura general.

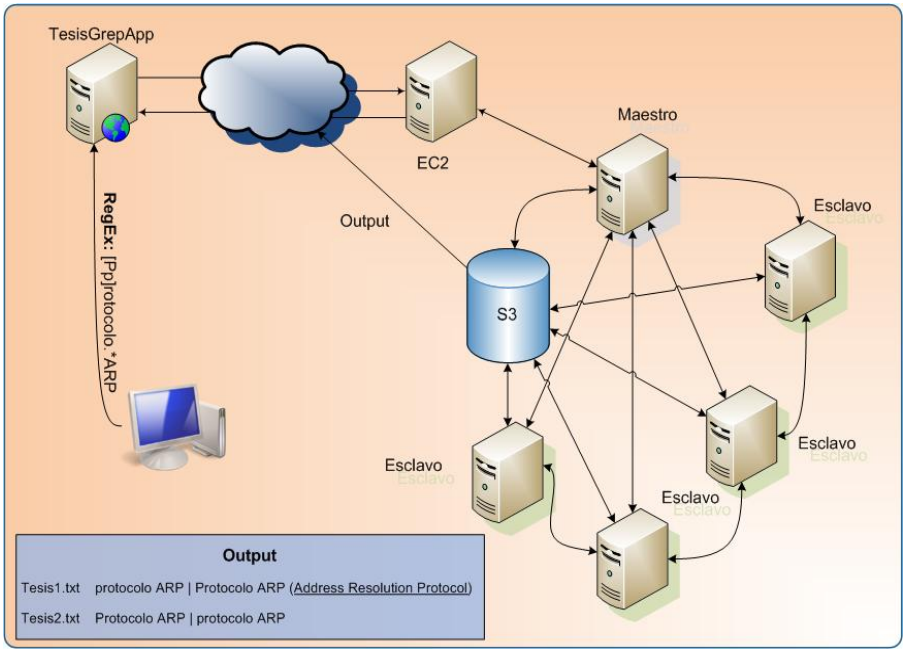


Gráfico 8: Arquitectura general del sistema

3.4.1. Pre-procesamiento de datos

Inicialmente los archivos digitalizados de las tesis se encontraban en formato PDF[1], para lo cual se vio la necesidad de someterlos a una etapa de pre-procesamiento con la finalidad de realizar la conversión de los mismos a texto plano (*.txt).

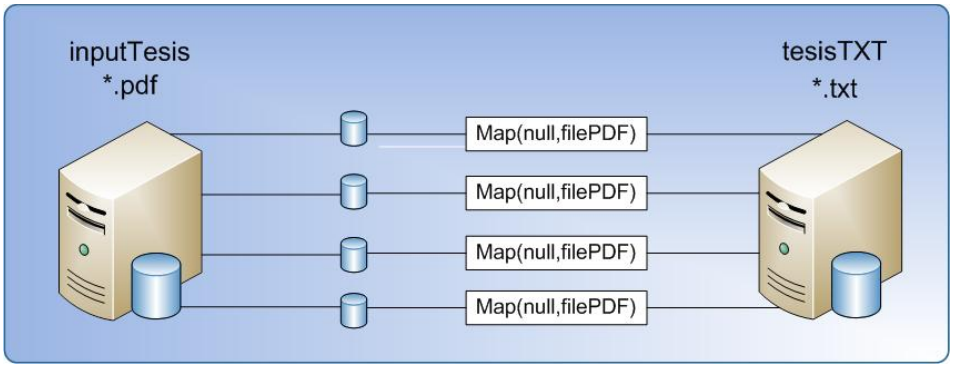


Gráfico 9: Proceso MapReduce para la conversión de archivos de entrada

Para la conversión se utilizó un proceso MapReduce sencillo como se indica en el gráfico 9. Por defecto el formato de entrada de los datos que se analizan en la fase map permite el procesamiento línea a línea, sin embargo para efectos prácticos fue modificado para recibir en cada map un archivo PDF completo, para lo cual utilizamos la clase WholeFileInputFormat como lo sugiere en (4).

[1] El Portable Document Format (PDF) es un formato de almacenamiento de documentos soportado por los sistemas operativos Windows, Unix/Linux y Mac. Un documento PDF tiene la misma apariencia, color, tipo de imprenta, gráficos y formato que un documento impreso. Fue desarrollado por la empresa Adobe Systems, solucionando problemas de compatibilidad de versiones o de aplicaciones.

En la fase Map**,** luego de recibir cada archivo se utilizó la librería PDFBox (5) 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 el TextInputFormat.

No fue necesario el uso de una fase reduce puesto que no se realizó ningún análisis sobre el contenido de los archivos.

3.4.2. Proceso MapReduce 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 el gráfico 10.

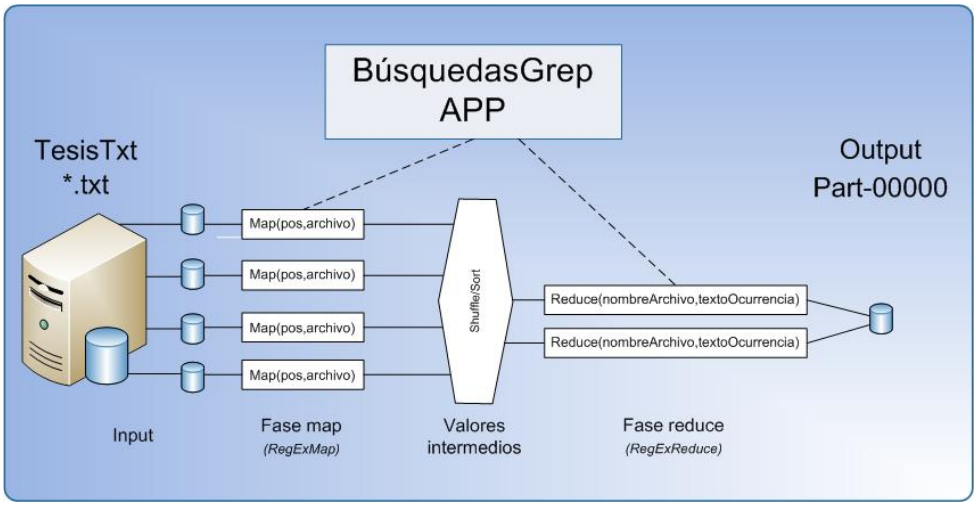


Gráfico 10: Proceso MapReduce para la búsqueda tipo Grep

- **RegExMap**

La función map para nuestro proyecto trabaja sobre tres parámetros, los cuales se describen a continuación:

WriteableComparable key - Byte de posición de inicio de línea.

Text value - Línea de un archivo del directorio de entrada.

OutputCollector - Permite generar los pares <clave, valor>.

El sistema de Hadoop divide el contenido del directorio de entrada en registros lógicos, por lo general, en líneas, luego se invoca al método map().

Nota: *En nuestro caso cada archivo contiene una única línea debido a que su contenido está escrito sin utilizar saltos de línea.*

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. Generando de esta forma pares <clave, valor>, en donde la clave corresponde al nombre del archivo y el valor contiene el texto encontrado que corresponde a la expresión regular ingresada por el usuario.

- **RegExReduce**

En la función reduce se concatenan todas las ocurrencias encontradas correspondientes al mismo archivo. Generando así, los pares <nombreArchivo1, ocurrencia1 | ocurrencia2 |…>.

Por ejemplo, para la expresión [Pp]rotocolo\s*ARP, la salida de la función Reduce sería la siguiente:

Tesis1.txt	protocolo ARP | Protocolo ARP
Tesis2.txt	Protocolo ARP | protocolo ARP

Nótese que se ha realizado una validación previamente para que las ocurrencias aparezcan solo una vez aunque en el documento se repitan varias veces.

CAPÍTULO IV

4 IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se puntualizan aspectos de implementación, entre ellos, la plataforma, librerías y servicios Web utilizados. Adicionalmente se encuentran detalladas las pruebas que se realizaron, así como su respectivo análisis de resultados.

4.1. Herramientas de desarrollo utilizadas

Para la implementación del Sistema de búsquedas avanzadas se utilizó el framework MapReduce del proyecto Hadoop de código abierto desarrollado por Apache, en su versión 0.18.3. Para pruebas locales se trabajó con una distribución de entrenamiento de Cloudera para Hadoop (6), para visualizarla se utilizó el programa VMware Player (7), versión 2.5.2 build-156735.

Al momento de ejecutar la aplicación de manera distribuida (8), 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* (9).

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 (10) y el clúster de tamaño variable que ofrece la capacidad de procesamiento para el sistema es provisto por el servicio EC2 (11).

Para el pre-procesamiento de los documentos se utilizó la librería PDFBox de Apache, versión 0.7.3, por medio de la cual se convirtieron los datos de entrada en formato PDF a formato de texto plano txt.

Además, como referencia, se revisó el código de un grep distribuido contenido en la librería de ejemplos de Hadoop “hadoop-0.18.3-patched-examples.jar”, a partir del cual se crearon las funciones RegExMap y RegExReduce que describen el proceso mapReduce del Sistema de búsquedas.

4.2. Inconvenientes durante el proceso de conversión

Después de realizar la conversión de los 3202 archivos correspondientes a los 13 GB de documentos de tesis en formato PDF se encontraron archivos que no pudieron convertirse correctamente, muchos de ellos se crearon como archivos en blanco (tamaño 0 KB). Esto se debe a que en el momento de la digitalización de ciertos documentos realizada por el Centro de Información Bibliotecaria no se utilizó un software con soporte de OCR (Reconocimiento óptico de caracteres) (12), obteniendo como resultado documentos cuyo contenido son imágenes en lugar de texto, lo cual provocó que la librería PDFBox no haya podido extraer el contenido de dichos archivos.

En el gráfico 11 se puede visualizar que el porcentaje de pérdida de datos que se obtuvo luego del preprocesamiento de las tesis de la ESPOL fue 16%. Es decir que para el Sistema de Búsquedas Grep se va a trabajar con 2683 archivos (Tamaño del dataset 429 MB).

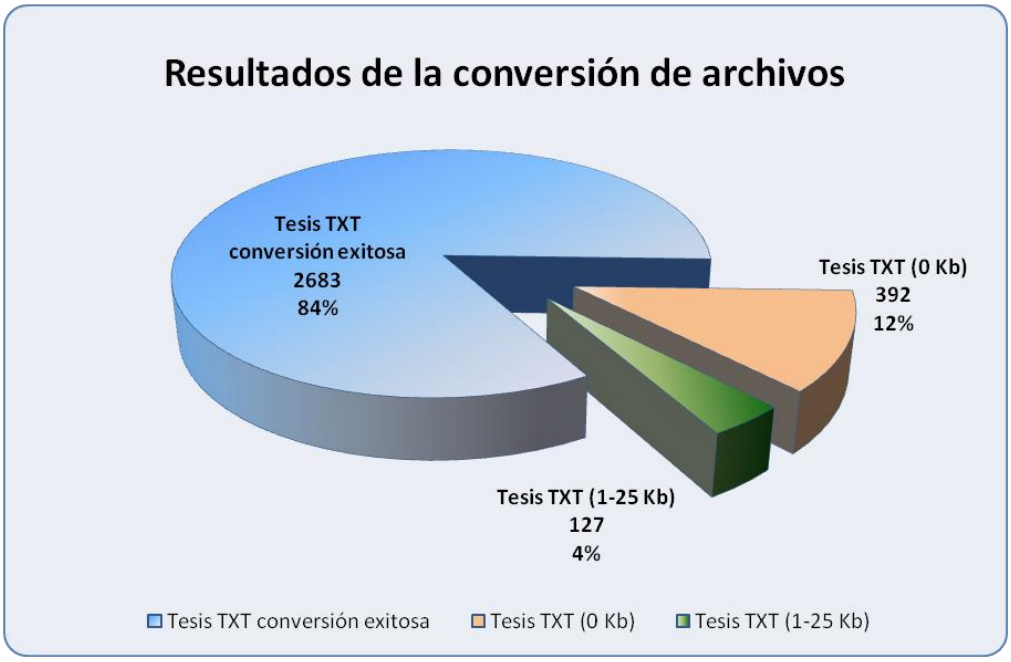


Gráfico 11: Resultados de la conversión de archivos PDF a TXT

4.3. Costo de los servicios Web de Amazon

Para la realización de este proyecto de graduación se obtuvo acceso gratuito a los Servicios Web de Amazon como se explicó en la sección 4.1 de este documento. Sin embargo, los costos reales de estos servicios se detallan en la tabla 2.

AWS	Costo	Detalle
S3 (13)	$0.15/Gb	Si sobrepasa de 50 Tb por mes el precio disminuye.
EC2 (14)	$0.20/Hora	Costo=$0.20*#Nodos

Tabla 2: Precio de los Amazon Web Services

Adicionalmente Amazon cobra $ 0.10 por GB transferido hacia el clúster y $ 0.17 por GB transferido desde el clúster.

Nota:

La transferencia de datos entre Amazon EC2 y Amazon S3 no tiene precio adicional.

4.4. Pruebas de rendimiento

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.

A continuación se adjunta la tabla de resultados de las pruebas realizadas.

Número de nodos	Dataset		Expresión regular	Tiempo de respuesta (Horas)
	Tamaño (MB)	N° archivos		
10	429	2683	"P(etroecuador|ETROECUADOR|etroEcuador)"	0:14:46
10	429	2683	"([Dd]ise.o|[Cc]onstrucci.n|[Aa]largamiento) de ([Ee]mbarcaci.n|[Bb]arco|[Ll]ancha|[Bb]uque)"	0:15:04
10	429	2683	"[Aa]n.lisis\s[Ee]stad.stico\sde.{1,40}"	0:15:01
10	429	2683	"[Ee]staci.n(.?|es)\sde\s[Rr]adio(difusi.n|comunicaci.n|(\s)?)"	0:14:50
10	429	2683	(?i)(an.lisis|dise.o|implementaci.on)([0-9a-zA-Z\s]+)?aplicaci.n([0-9a-zA-Z\s]+)?(.?multimedia|interactiva(s?))(\s?)	0:19:22
12	429	2683	"[Aa]plicaci.n(es)?\s([Ii]nteractivas?|[Ww]eb\s[Dd]in.micas?)"	0:11:38
12	429	2683	"(?i)([Aa]plicaci.n(es)?\s([Ii]nteractivas?|[Ww]eb\s[Dd]in.micas?))"	0:11:35
12	429	2683	"(?i)([Dd]esarrollo|[Aa]ctividad)\s[Tt]ur.stic[oa]"	0:12:11
12	429	2683	"[Pp]ropietario.{0,16} ([Cc].digo [Aa]bierto|[Ss]oftware\s[Ll]ibre)"	0:12:16
12	429	2683	"[Pp]roducci.n\sde\s[Ee]nerg.a\s[Ee]l.ctrica"	0:12:11
15	429	2683	(?i)[Gg]esti.n\s+de\s+[Pp]royecto(s)?\s+[Ii]nform.tico(s)?	0:09:14
15	429	2683	(?i)(voz\ssobre\sip|((comunicaci.n(es)?|tecnolog.as?|arquitecturas?)(de)?)voip)	0:09:05
15	429	2683	([Pp]lan|[Ee]studio) de [Mm]ercade?o	0:09:27
15	429	2683	(?i)(sistema de informaci.n geogr.fico.{0,30})	0:10:42
15	429	2683	(?i)(desarrollo([0-9a-zA-Z\s]+)?plan\s+de\s+marketing)	0:09:33
18	429	2683	[Pp]rotocolo\s*ARP	0:07:40
18	429	2683	\.([0-9a-zA-Z\s])+[Ee]stero [Ss]alado([0-9a-zA-Z\s])*\.	0:08:01
18	429	2683	INFLUENCIA\s+DEL\s+NIVEL\s+DE\s+RIESGO\s+POL.TICO\s+Y\s+ECON.MICO\s+EN\s+EL\s+VOLUMEN\s+DE\s+INGRESO\s+DE\s+TURISTAS\s+EN\s+EL\s+ECUADOR	0:07:47
18	429	2683	([Aa]n.lisis|[Dd]etermina(r|ci.n)) (de|los) [Rr]iesgos?([Ff]inancieros?)?	0:07:50
18	429	2683	[Ss]istemas? (de [Pp]osicionamiento [Gg]lobal|GPS)	0:07:34
20	429	2683	[Vv]irus VIH|VIH\SSIDA	0:08:48
20	429	2683	(?i:((estrategia(s)?|t.cnica(s)?)([0-9a-zA-Z\s]*)inteligencia artificial))	0:07:21
20	429	2683	[Cc]entral(es)? ([Hh]idroel.ctricas?|[Tt]ermoel.ctricas?) .{0,30}	0:08:25
20	429	2683	([Ii]ncrement(o|a|ar)|[Aa]ument(o|a|ar)) (de)?la [Pp]roductividad	0:07:09
20	429	2683	[Aa]n.lisis\s+y\s+([Dd]ise.o|[Cc].lculos?)\s+[Tt].rmicos?	0:06:55

Tabla 3: Resumen de pruebas de rendimiento realizadas

5 ANÁLISIS DE RESULTADOS

5.1. Pruebas de rendimiento

Se tomaron cinco mediciones de tiempo de respuesta para cinco tamaños distintos del clúster de EC2, los resultados serán analizados a continuación.

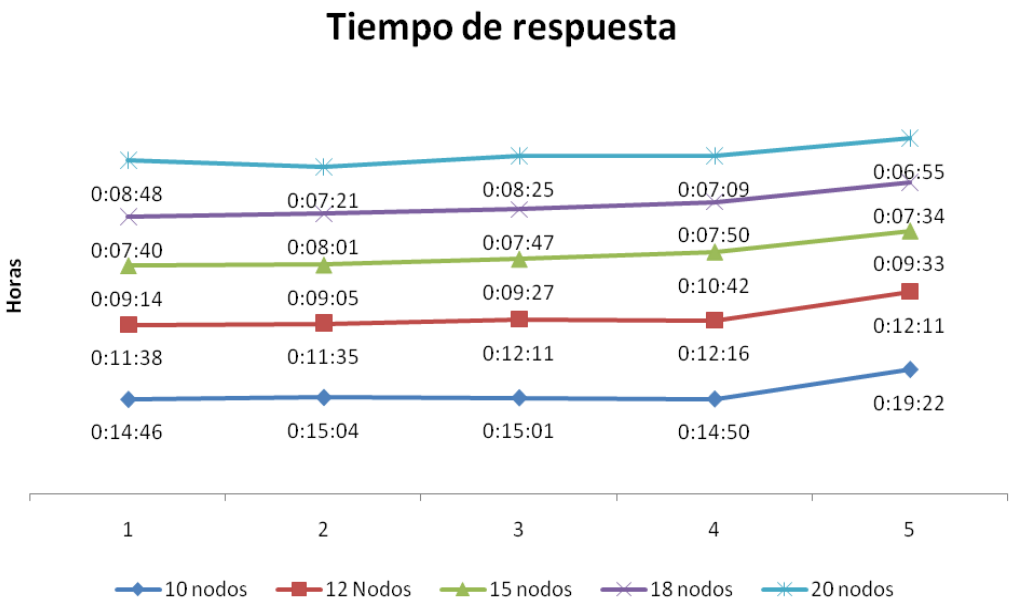


Gráfico 12: Tiempos de respuesta

En el **Gráfico 12** 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.

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

# 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

Tabla 4: Tiempo promedio de ejecución

Consideramos que un tiempo de respuesta aceptable sería de cinco minutos, sin embargo con el valor máximo de nodos disponibles para realizar nuestras pruebas no logramos obtenerlo. Estimamos que con un clúster de 25 nodos podríamos llegar a tiempos de respuesta mejores.

5.2. Costos del sistema

De acuerdo a la Tabla **2** relacionada con los precios del los Servicios S3 y EC2 de Amazon se ha calculado el costo real de la conversión (PDF - TXT) de las tesis de la ESPOL y de la ejecución de una consulta en el Sistema de Búsquedas tipo Grep de acuerdo al número de nodos del clúster utilizados.

Las fórmulas utilizadas para calcular los costos son las siguientes:

COSTO EC2 = Costo de procesamiento en EC2 * # nodos en clúster * # de horas en que el clúster estuvo levantado (se redondean al entero superior) + $ 0.10 por GB transferido hacia el clúster + $ 0.17 por GB transferido desde el clúster

COSTO S3 = Tamaño del dataset * Costo del almacenamiento en S3

Costo Total = Costo EC2 + Costo S3

5.2.1. Costos de la conversión

Número de Nodos levantados en el clúster	20
Número de horas necesarias para la conversión	1
Tamaño de datos a convertirse - GB	13
Costo de Almacenamiento en S3 - USD/GB	0,15
Costo de Procesamiento en EC2 - USD/Hora	0,20
Transferencia hacia el clúster del código de conversión (jar) - USD/GB	0,10
Transferencia desde el clúster de los archivos convertidos - USD/GB	0,17
Total S3	1,95
Total EC2	4,27
Precio Total de Conversión	**$ 6,22**

Tabla 5: Precio del proceso de conversión PDF-TXT

El costo de la conversión de 13 GB en documentos de tesis de la ESPOL utilizando un clúster de 20 nodos sería $6.22, sabiendo que el tiempo aproximado que se toma el programa de conversión para esta cantidad de documentos es de una hora. Este valor es bajo considerando que esta cantidad de tesis se convierten a texto plano una sola vez y posteriormente se trabaja para el sistema de búsquedas tipo Grep con los documentos convertidos, cuyo tamaño se redujo a tan sólo 429 MB.

Dado que el sistema de búsquedas tipo Grep es escalable, cada seis meses podrían convertirse las tesis nuevas realizadas en la universidad sin afectar al funcionamiento de esta aplicación. Si se estima que el tamaño total de las tesis cada seis meses sería mucho menor a 1 GB, se podría decir que el precio aproximado de su conversión utilizando sólo 10 nodos sería tan sólo de $2.42.

Número de Nodos levantados en el clúster	10
Número de horas necesarias para la conversión	1
Tamaño de datos a convertirse - GB	1
Costo de Almacenamiento en S3 - USD/GB	0,15
Costo de Procesamiento en EC2 - USD/Hora	0,20
Transferencia hacia el clúster del código de conversión (jar) - USD/GB	0,10
Transferencia desde el clúster de los archivos convertidos - USD/GB	0,17
Total S3	0,15
Total EC2	2,27
Precio Total de Conversión	**$ 2,42**

Tabla 6: Precio estimado de la conversión semestral

5.2.2. Costos de la ejecución de una consulta

El costo del almacenamiento de las tesis en formato TXT en el servicio S3 de Amazon es de $0,15 por mes, puesto que el tamaño almacenado es de 429 MB (para los cálculos se redondea a 1 GB).

En la **Tabla 7** se detallan los costos del Servicio EC2 que equivalen a una consulta en el sistema de búsquedas tipo grep de acuerdo a la cantidad de nodos utilizados en el clúster para el procesamiento.

Número de Nodos levantados	10	12	15	18	20
Costo de Procesamiento en EC2 - USD/Hora	0,20	0,20	0,20	0,20	0,20
Transferencia hacia el clúster del código Grep (jar) - USD/GB	0,10	0,10	0,10	0,10	0,10
Transferencia desde el clúster del archivo de salida - USD/GB	0,17	0,17	0,17	0,17	0,17
Total EC2	**2,27**	**2,67**	**3,27**	**3,87**	**4,27**

Tabla 7: Costos de ejecución de una consulta

5.3. Tiempos de respuesta Vs costos

Variando la cantidad de nodos de 10 a 20 se comprobó que el costo incrementa mientras el tiempo de respuesta disminuye. Esto se puede visualizar en el **Gráfico 13**.

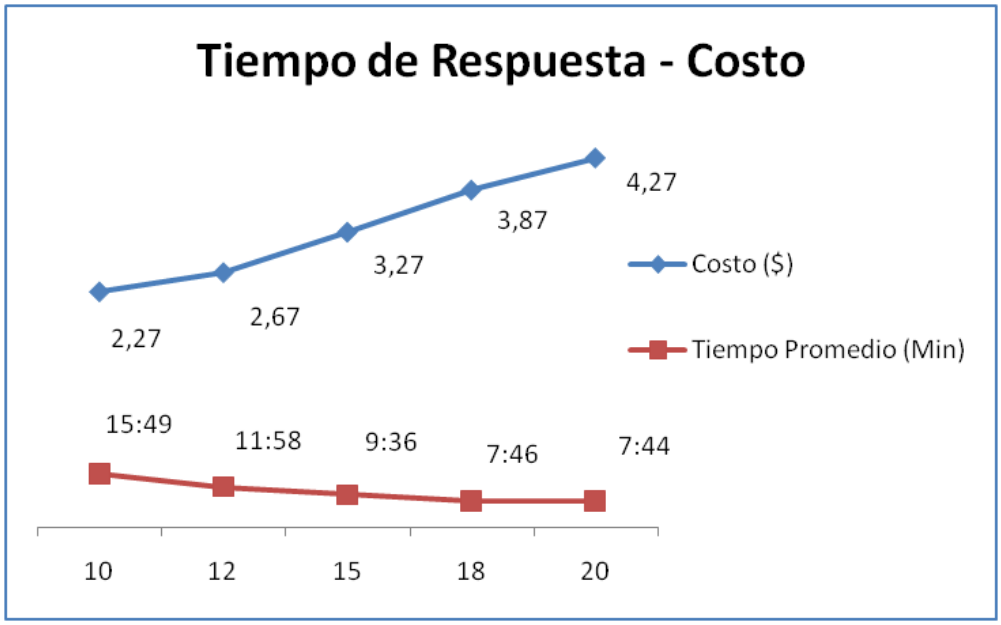


Gráfico 13: Tiempos de respuesta promedio y costos del sistema

CONCLUSIONES

1. 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.

2. Los tiempos de respuesta obtenidos con el uso máximo de los recursos disponibles fueron en promedio 7 minutos con 44 segundos. Es un tiempo prudente considerando que las búsquedas se realizan en tiempo real dentro de todo el contenido de tesis disponibles para este sistema, lo cual constituye alrededor de 2700 archivos analizados en cada consulta.

3. 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.

4. El fácil acceso a servicios rentados que ofrecen tamaños variados de capacidad de procesamiento y almacenamiento como los Amazon Web Services han permitido a las empresas explorar nuevas tecnologías sin necesidad de incrementar sus equipos, software o aumentar personal, lo que representa inmensos ahorros, con el consiguiente beneficio para la sociedad.

RECOMENDACIONES

1. Los tiempos de respuesta pueden mejorar significativamente con el análisis de un número más óptimo de mappers y reducers para el sistema. Nuestras pruebas se realizaron utilizando nueve mappers y un reducer, valores que fueron establecidos por defecto, sin ninguna investigación preliminar.

2. 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.

3. 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.

4. Sería recomendable considerar implementar el sistema en clúster propio, para no tener que incurrir en costos por consulta. Esto se podría hacer en el futuro Parque del Conocimiento de la ESPOL, que contará con un data center propio

REFERENCIAS BIBLIOGRÁFICAS

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

4. White, Tom - O'Reilly. *"Hadoop: The Definitive Guide".* Junio 2009.

5. The Apache Software Foundation. [En línea] [Citado el: 3 de Octubre de 2009.]

http://incubator.apache.org/pdfbox/download.html#pdfbox.

6. Cloudera. [En línea] [Citado el: 3 de Octubre de 2009.] http://www.cloudera.com/hadoop.

7. VMware. [En línea] [Citado el: 3 de Octubre de 2009.]

http://www.vmware.com/products/player.

8. Wikipedia. [En línea] [Citado el: 3 de Octubre de 2009.]

http://en.wikipedia.org/wiki/Cloud_computing.

9. Amazon. [En línea] [Citado el: 3 de Octubre de 2009.] http://aws.amazon.com/education/.

10. Amazon. [En línea] [Citado el: 3 de Octubre de 2009.] http://aws.amazon.com/s3.

11. Amazon. [En línea] [Citado el: 3 de Octubre de 2009.] http://aws.amazon.com/ec2.

12. Wikipedia. [En línea] [Citado el: 3 de Octubre de 2009.]

http://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres.

13. Amazon. [En línea] [Citado el: 3 de Octubre de 2009.]

http://aws.amazon.com/s3/#pricing.

14. Amazon. [En línea] [Citado el: 3 de Octubre de 2009.]

http://aws.amazon.com/ec2/#pricing.