Tecnológico de Costa Rica

TEC

Mini Buscador Web

Arquitectura y Diseño

Bryan Steve Jiménes Chacón – 2014114175
Roberto Fernández Valerio - 2013010399

Contenido

Introducción

La tecnología de hoy en día el auge del internet han provocado un gran aumento en la cantidad de datos que se reciben, estos datos se presentan en cantidades tan grandes que contienen más información de la que se puede procesar normalmente, por esto se han creado nuevas estrategias para poder extraer información y datos útiles, como la tecnología de hadoop que implementa una forma de procesamiento map-reduce, que busca encontrar datos importantes en una gran cantidad de datos a partir de ciertos parámetros clasificando los datos y luego analizándolos según lo que se busque encontrar.

WebCrawler

Para crear obtener la información que posteriormente se procesara con hadoop se utilizará php, el código comienza inicializando una serie de arrays que nos permitirán guardar y discriminar la información importante en las páginas web, luego se comienza con la función crawlPage que recibe tres parámetros, la url, el archivo y la profundidad para la recursión, además de las variables definidas al principio se crea un array para las urls visitadas y otra para el contenido de la url. Utilizamos la función fwrite de php para escribir en un documento la información según lo que se reconozca, ya sea la url, el contenido, los tags o el título de la página con su respectivo delimitador y por medio de un foreach y de condiciones creamos la recursión hasta que la profundidad que hayamos definido llegue a 0, finalmente se retorna el resultado.

Para cada sección de la información se utilizó una función por separado las cuales se explicarán a continuación, la función get_content es de las más importantes ya que recibe el contenido de una url le da formato al texto eliminando los tag y los símbolos innecesarios por medio de los delimitadores previamente definidos al inicio del php y devuelve el resultado. La funcion page_title busca una coincidencia con un url, y si no lo encuentra guarda como resultado que no tiene título, pero si lo encuentra le da formato al link y lo devuelve en un return. Además tenemos la funcion get_tags que recibe el contenido de la página, y tiene una variable local donde se dan formato a los tag y se guardan en una variable que es devuelta por medio de un return.

Finalmente, todo el crawler se comienza a ejecutar cuando creamos un documento que podamos editar con la función fopen, luego definimos una lista de urls en forma de array, los cuales serán procesados por la función crawlPage del principio del documento con un link como parámetro y el documento por editar, esto va a crear la recursión para cada

link y los links que contengan los primeros links. Cuando todos los links han sido procesados simplemente se procede a utilizar la función fclose para cerrar el documento que estamos escribiendo y que hadoop lo pueda procesar

Hadoop, MapReduce y Mysql

Para utilizar hadoop y analizar el Big Data que se extrajo con el web crawler se utilizó la máquina virtual de cloudera, esto porque ya tiene el ambiente montado y también posee una base de datos MySQL ya instalada por defecto.

Para el primer job primeramente se importaron las librerías necesarias para utilizar hadoop y el map-reduce, se creó una clase llamada simplemente Job1 con un logger para el registro de los eventos. El main del job incluye una llamada a un método llamado run que es el que se encarga de cargar la configuración para el job, las clases que se van a ocupar, los input y output donde se van a leer y guardar los datos y el tipo de dato con el que se creará el output.

El método Map es el encargado de leer los datos, y guardarlos como Text para el Reduce, para poder clasificar lo que se está leyendo se utilizan los delimitadores que se definieron previamente en el crawler para la url, el texto, tag, y título de la siguiente manera, se empieza a leer cada línea de los datos, según el delimitador se le asigna a una variable si es url, tag o título, si no es ninguna de las anteriores es porque es texto con lo que se guarda la palabra que se leyó como llave y la url como valor de la llave, el proceso se repite hasta terminar el documento, luego el Reduce recibe y retorna datos tipo Text, el Reduce es muy sencillo simplemente recorre los resultados por cada vez que la palabra aparezca se agrega en un StringBuilder llamado links el link, cuando termina hace un write de la palabra como llave y de los links como tipo Text, esto hasta terminar de

analizar los resultados. Dichos resultados se guardan en la carpeta output creada para el job en cuestión.

El segundo Job2 inicia similar al primero, se importaron las librerías de hadoop y luego en el main también se carga la configuración y los parámetros correspondientes. El Map del segundo Job2 tiene un proceso de lectura similar donde se busca el delimitador y según el resultado se clasifica si es url, tag o título, sin embargo, en la escritura si tiene un proceso diferente, como se deben analizar las palabras repetidas por cada sitio web se decidió utilizar como llave una concatenación de la palabra en cuestión junto con el url, y como valor una variable final de tipo IntWritable con el valor de 1 para indicar que se encontró “x” palabra en “y” sitio web. Por su parte, el Reduce recibe variables tipo Text e IntWritable en el Reduce se definió una variable sencilla de tipo int como contador llamada sum, esta variable, inicializada en 0, es incrementada en uno por cada vez que encuentre una llave igual, finalmente solo se hace un write con la llave, es decir, la palabra y la url con la variable sum como valor, la cual corresponde al total de repeticiones.

El tercer Job tiene una estructura más simple, primeramente importamos de nuevo las librerías y creamos una clase llamada Job3, de la misma manera el contiene los argumentos y la configuración del job, seguidamente en el Map nos damos a la simple tarea de guardar cada palabra como llave y en el valor asignarle una variable final tipo IntWritable con el valor de 1, este proceso se repite hasta terminar el documento, el Reduce recibe datos de tipo Text e IntWritable y retorna los mismos tipos, su función es la misma que en el job anterior pero implementado un poco diferente para practicar el uso de hadoop, en este caso, declaramos una variable de tipo int llamada sum, en el ciclo for que recorre los resultados del Map por cada palabra (llave) que se repita se utiliza el método get y se le suma al count el valor, recordemos que siempre le asignamos 1 como valor a todas las llaves, por lo que el resultado final será el total de repeticiones

de la palabra, y solo resta utilizar el método write con la palabra como llave y la variable sum como valor para guardar los resultados en el output de hadoop.

Aprovechando la máquina virtual de cloudera usamos su base de datos para guardar los resultados de los jobs. Modelamos tres tablas, una para cada job, con los campos correspondientes dígase la palabra, urls, conteos, etc. Y creamos un programa en java que importa, aparte de las librerías de hadoop, las de MySql con lo que procedimos a crear métodos para conectarse, ejecutar sentencia y realizar consultas fácilmente, con estos métodos listos en el main realizamos la conexión a la base de datos primero que nada, seguidamente creamos variables de hadoop con la configuración y la ubicación del resultado del map-reduce, para leerlos usamos un BufferedReader con todo lo necesario para que puede leer el archivo, creamos una variable line de tipo String para guardar los datos, y comenzamos a leer línea por línea los resultados, cada línea se divide en una array utilizando intencionalmente un espacio en blanco como delimitador en los resultados de hadoop y se asignan las partes del array a las variables que corresponden según el job, con estas variables guardadas procedemos a ejecutar un sql de inserción en la tabla correspondiente al job que estamos procesando y lo ejecutamos como nuestro método ejecutar, este proceso se repite mientras la línea que se lea no tenga valor de null, con esto tenemos los datos insertados en la base de datos y listos para ser consultados desde nuestro buscador.

Buscador

Para la búsqueda de una o varias palabras, se utilizó un form en html, validando los campos si cumplen los requisitos (que no sean vacíos, o que solo sea un caracter), y luego invoca una función PHP por medio del método POST.

El buscador utiliza se utilizaron las librerías de conexión con MySQL, el cual es necesario para un correcto funcionamiento entre Apache y la base de datos.

Para poder realizar las consultas a la base de datos, se debió establecer de manera anticipada un connection string, ingresando el servidor, usuario, contraseña y nombre de la base de datos a consumir.

Cuando este establece la conexión, valida los campos ingresados anteriormente, y realiza un ciclo en el cual crea un string de consulta para enviárselo a la base de datos. Con los resultados dados, estos se muestran en una nueva página, mostrando los links y el título del mismo.

Si es más de una palabra a consultar, se podrá ver las separaciones entre las palabras ingresadas en el form. Dependiendo de la consulta, esta puede tardar un poco más que otras.

Conclusión

Como pudimos ver, hadoop ofrece muchas posibilidades para el análisis rápido de grandes cantidades de datos, la idea de este proyecto fue la de demostrar cómo es que el manejo de hadoop facilita la manipulación de datos y pone al alcance del programador aplicaciones que de otro modo serían imposibles de realizar sin un desperdicio enorme de tiempo y de recursos, aunque su curva de aprendizaje pueda ser más alta que realizar un programa normal debido a las partes “oscuras” de su código, donde no se tiene conocimiento de lo que sucede explícitamente, podemos tener confianza de que los algoritmos son correctos y que el funcionamiento de hadoop está bien optimizado debido a que tiene apoyo por parte de Google y una vez que se domina esta parte se pueden realizar todo tipo de análisis y de aplicaciones que manejan inclusive millones de datos alrededor del mundo .