ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
CENTRO DE INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA

Implementación de un módulo de búsqueda de personas dentro de una base de datos de rostros en un ambiente distribuido usando Hadoop y los Servicios Web de Amazon (AWS).

Ángel Merchan, Juan Plaza y Juan Moreno
Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del litoral (ESPOL)
Campus Gustavo Galindo, Km. 30.5 vía Perimetral
Apartado 09-01-5863. Guayaquil, Ecuador
angel.merchan@gmail.com, jplazaarguello@gmail.com, jmoreno@fiec.espol.edu.ec

Resumen

El procesamiento digital de imágenes toma cada día más importancia. Generalmente los algoritmos utilizados para el procesar éstas imágenes son de complejidad computacional alta por la cantidad de datos a procesar y las operaciones que se hacen sobre los mismos. Por esta razón muchas de estas operaciones que involucran cálculos con matrices y vectores, son llevadas a hardware en las tarjetas gráficas para no cargar al procesador central (CPU) e incrementar el rendimiento.

El presente trabajo consiste en realizar la implementación de un módulo para la búsqueda de personas en una base de datos de rostros por medio del reconocimiento facial .

En el desarrollo del mismo se aprovechará las bondades del procesamiento distribuido proporcionado por el modelo de programación MapReduce e implementado en la plataforma hadoop.

Palabras claves: *MapReduce, Hadoop, HDFS, AWS, luxand, template, buffer, STL.*

Abstract

The digital processing of images take more importance every day. Generally, the algorithms used for processing these images are of high computational complexity caused by the amount of data to process and operations executed on them. For this reason many of these operations involving matrix and vectors calculations, are carried to hardware in the graphics card to not load the processor unit (CPU) and increase performance.

This work consists about implementing a module to search for people in a database faces through facial recognition.

In its development, we will take advantage about the benefits of distributed processing, provided by the MapReduce programming model and implemented in the Hadoop platform.

1. Introducción

La masificación en la digitalización de todo tipo de información en las últimas dos décadas ha creado un ambiente adecuado para que la investigación en las tecnologías de la información se diversifique. En el caso de las imágenes no ha sido la excepción. Con la llegada de las cámaras digitales a los inicios de los noventa y luego con su popularización a consumidores de casa a fines de la misma década, se hicieron avances significativos en el campo del procesamiento de imágenes.

El procesamiento digital de imágenes toma cada día más importancia y las aplicaciones que se le pueden dar son diversas. Las tecnologías en ésta área se han perfeccionado, en donde la inteligencia artificial ha sido de importante ayuda para el desarrollo de las mismas.

Generalmente los algoritmos utilizados para el procesamiento de imágenes son de complejidad computacional alta por la cantidad de datos a procesar y las operaciones que se hacen sobre los mismos. Por esta razón muchas de estas operaciones que involucran cálculos con matrices y vectores, son llevadas a hardware en las tarjetas gráficas para no cargar al procesador central (CPU) e incrementar el rendimiento.

Sabiendo esto decidimos implementar un módulo que aprovechara los beneficios del procesamiento en paralelo para obtener resultados en mucho menos tiempo que utilizando un computador. Además de la capacidad que tendría para procesar cientos de miles de archivos; lo que en un único computador no se podría.

2. Análisis del Problema

En lo que se refiere a este trabajo nos enfocaremos en la parte de reconocimiento y detección facial. Aunque los métodos utilizados siguen sin ser perfectos, han mejorado notablemente desde que empezaron a ser utilizados. Estas técnicas están enfocadas a aplicaciones de seguridad y entretenimiento principalmente.

El reconocimiento de personas mediante el procesamiento de imágenes no es una tecnología muy utilizada en la actualidad. Aún se está realizando mucha investigación en este campo debido a que estas técnicas no son muy precisas. Sin embargo algunos países han tomado la iniciativa de ponerla a prueba; por ejemplo, Australia con su sistema SmartGate, instalado desde el 2007, agilita el proceso de los pasajeros que ingresen a dicho país y al mismo tiempo verifica que el portador del documento sea realmente el dueño del mismo. El departamento de aduana del Reino Unido también decidió implementar un sistema similar de escaneo facial en los aeropuertos desde el verano del 2008. Este tiene el mismo propósito que el anterior, agilitar el proceso de aduana y hacer revisiones de seguridad.

3. Objetivos

Uno de los objetivos es demostrar que Hadoop además de ser una plataforma eficiente para el procesamiento masivo de archivos de texto, también lo es para el procesamiento masivo de imágenes.

Adicionalmente queremos comprobar que implementar una solución de procesamiento masivo de imágenes resulta sencillo y muy poco costoso.

4. Detección Facial

La detección facial es un método de computación cuyo objetivo es determinar si en una imagen dada existen o no rostros. De estar presentes, retorna la posición y dimensiones para cada rostro. Existen varias soluciones para resolver este problema, pero en general pueden ser agrupadas como tareas de clasificación de patrones; esto es, se extraen características que luego son comparadas con un modelo establecido [1]. Dependiendo del método a utilizar este modelo puede ser construido mediante entrenamiento o ser establecido por reglas fijadas por el investigador; aunque el último es muy poco utilizado [2].

En el método propuesto por Viola y Jones el entrenamiento del clasificador se hace tomando como base características tipo Haar (Haar-like features) que describen la clase del objeto para la que se hará el entrenamiento [3]. Rowley y compañía [4] desarrollaron un sistema de detección de rostros basado en redes neuronales. El proceso consiste de 2 etapas:

La primera consiste en un filtro basado en una red neuronal, a la cual se le proveen imágenes de tamaño fijo desplazando ventanas de tamaño arbitrario por cada punto de la imagen en varias repeticiones. Para cada imagen el filtro genera una salida entre 1 y -1 que significan la presencia o ausencia de un rostro, respectivamente.

La segunda fase se encarga de eliminar las ventanas que hayan sido identificadas erróneamente como caras. Los rostros son detectados usando diferentes tamaños de ventanas, por lo que basta fusionar los resultados obtenidos alrededor de una posible cara para obtener una respuesta. Un método heurístico es utilizado para descartar los falsos positivos. Esta regla heurística se la obtiene de la observación; sabiendo que un rostro es detectado por varias ventanas de diferente escalas en su vecindad, si el número de detecciones está por debajo de un determinado umbral entonces la imagen no es considerada como un rostro.

5. Reconocimiento Facial

El reconocimiento facial consiste en identificar o verificar una persona a partir de una imagen o un cuadro de video.

Uno de los métodos para realizar esto es mediante la comparación de distancias entre ciertas características faciales claves específicamente seleccionadas tales como ojos, nariz y boca, además el ángulo del maxilar, frente y distancias de varias porciones del rostro.

Incorporando todos estos datos numéricos obtenidos se crea una plantilla única, la cual es comparada con la plantilla de cada una de los rostros existentes en una base de datos de imágenes.

El reconocimiento facial es aplicable a seguridad y entretenimiento. Hoy en día es utilizado para tareas tales como conceder acceso a una computadora personal y hasta es utilizado en muchos aeropuertos en EEUU para la detección de sospechosos.

6. MapReduce

MapReduce es un modelo de programación introducido por google para brindar soporte al procesamiento en paralelo, es utilizado para escribir aplicaciones que procesen rápidamente grandes cantidades de datos en paralelo sobre grandes grupos de computadoras. Se especifica una función map que procesa un par clave/valor para generar una colección de pares clave/valor, y una función reduce que agrupa todos los valores intermedios asociados con la misma clave intermedia.

7. Hadoop

Apache Hadoop es un framework de código abierto hecho en java para ejecutar aplicaciones sobre grandes clústeres. Hadoop es un proyecto de Apache de alto nivel. Se basa en la colaboración de una extensa comunidad activa de contribuyentes alrededor del mundo para su éxito. [6]

Hadoop consiste del Sistema de Archivos Distribuidos de Hadoop (HDFS) y la implementación de la programación del paradigma Map-Reduce.

Este framework nos permite escribir y ejecutar aplicaciones que procesan una gran cantidad de datos.

Las principales características de Hadoop son: Escalable, económico, eficiente, confiable. [7]

7.1. Hadoop Pipes

Pipes es una librería que permite usar Hadoop DFS y poder escribir el Mapper y el Reducer en C++.

Las aplicaciones que requieren un ejecutar y procesar grandes operaciones matemáticas pueden tener un mejor rendimiento si es escrito en C++ y poder usar MapReduce a través de Pipes. Esta librería es soportada por sistemas Linux de 32 bits.

Hadoop Pipes convierte los datos a bytes que son enviados vía socket, tanto las claves como los valores son recibidas como STL strings (std::string).

Todas las funciones y clases de C++ están en el namespace de HadoopPipes, el trabajo puede consistir en la combinación entre C++ y Java de RecordReaders, Mappers, Partitioner, Combiner, Reducer, y RecordWriter. [8]

7.2. Sistema de Archivos Distribuido de Hadoop (HDFS)

El sistema de archivos distribuido de Hadoop es un sistema de ficheros distribuidos inspirado en el ya existente Sistema de Archivos de Google GFS y está diseñado para ejecutarse en hardware. Tiene muchas similitudes con los actuales sistemas de archivos distribuidos, sin embargo las diferencias con los otros sistemas de archivos distribuidos son significativas al contar con características implementadas para hacer frente a los problemas que se presentan en el almacenamiento y transferencias de archivos en un ambiente distribuido.

HDFS es altamente tolerante a fallos y está diseñado para ser implementado en hardware de bajo costo, proporciona acceso de alto rendimiento de datos de aplicación y es adecuado para aplicaciones que tienen grandes conjuntos de datos. HDFS está diseñado para soportar archivos de gran tamaño, los tamaños de bloques de archivos usado por HDFS es de 64 MB.

8. Amazon Web Services (AWS)

Amazon web Services es una infraestructura de servicios web en la nube. Con AWS se puede obtener poder de cómputo, almacenamiento y otros servicios, otorga la flexibilidad para elegir cualquier plataforma de desarrollo o modelo de programación. Amazon Web Services ofrece una serie de beneficios para las organizaciones de TI y desarrolladores entre ellos están:

Rentabilidad.- se paga solo los recursos que se usan sin compromisos por adelantado.

Fiabilidad.- Se utiliza una infraestructura web escalable, segura y resistente otorgando confiabilidad.

Completa.- Para no empezar desde cero AWS ofrece un sinnúmero de servicios que se pueden incorporar a las aplicaciones y las ayudan a que su construcción sea rentable y con menos inversión inicial.

8.1. Amazon Elactic Compute Cloud (EC2)

EC2 es un servicio que provee de capacidad de cómputo de tamaño variable en la nube. Presenta un entorno virtual lo que nos permite utilizar el servicio de interfaces web para poner en marcha las instancias, manejar los permisos de acceso de red y para configurar de manera simple la capacidad de recursos que necesitamos y obtener un control total de los mismos.

Para utilizar EC2 se debe seleccionar una imagen pre-configurada o crear una imagen de las máquinas virtuales de Amazon (AMI) que contiene aplicaciones, bibliotecas, datos y opciones de configuración.

9. Diseño e Implementación

9.1. Diseño General

En cuanto al entorno distribuido este módulo conserva el esquema de funcionamiento de un trabajo MapReduce convencional. La única diferencia de nuestro trabajo frente a los convencionales, es que debido a que los archivos que procesamos son imágenes, es necesario leer el archivo en su totalidad para que sean comprendidos por los algoritmos de detección y reconocimiento facial. Es por esto que utilizamos el WholeFileInputFormat que toma un archivo de entrada, lo lee en su totalidad y lo envía como valor a un mapper para su posterior procesamiento.

El WholeFileInputFormat difiere en gran medida de los formatos de entrada convencionalmente utilizados ya que estos leen secciones de los archivos para luego enviar cada pedazo de información a un mapper diferente.

Como se menciona anteriormente cada mapper recibe una imagen, la misma que analiza para reconocer el rostro dentro de la imagen. Luego se obtiene las características del rostro encontrado para luego compararlas

con las características del rostro de la persona que se busca. De esta manera se obtiene un valor numérico que denota el grado de similitud entre los rostros comparados. Este número es emitido hacia la fase reduce como valor dentro del par clave-valor (<key, value>) junto con el nombre del archivo de la imagen que es utilizada como la clave.

Luego de esto la fase de ordenamiento se encarga de listarlos ordenadamente para imprimirlos en el archivo de salida de resultados, mostrando así el archivo con mayor similitud a la imagen utilizada como entrada.

En la Figura 1 se muestra el esquema general del funcionamiento de nuestro módulo.

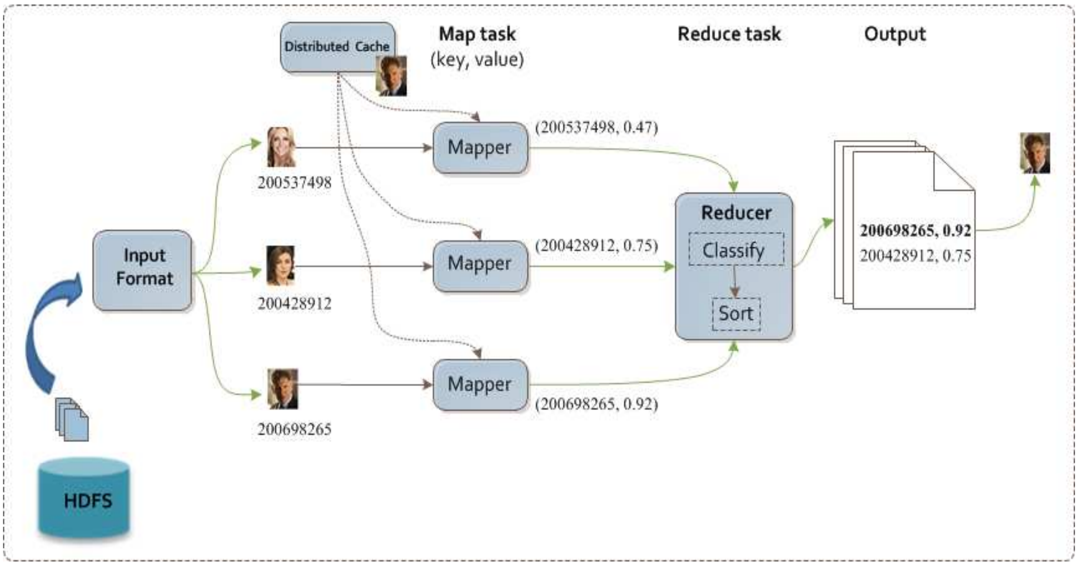

Figura 1: Arquitectura General del Módulo

9.2. Tecnología de Procesamiento de Imágenes

El presente trabajo trata de destacar y aprovechar las bondades que ofrece la plataforma Hadoop. Teniendo esto en cuenta decidimos que deberíamos utilizar una solución de detección y reconocimiento ya existente en el mercado en lugar de implementarla, debido a que sería una tarea tanto o más compleja que la implementación del presente proyecto.

El FaceSDK de Luxand nos provee de las funciones de procesamiento que necesitamos. Este SDK provee funciones que reconocen el rostro dentro de una imagen, luego extrae una plantilla y la representa en memoria como una cadena de caracteres. Las plantillas extraídas de cada imagen son comparadas mediante una función provista también por la librería, la misma que retorna un número decimal (float).

9.3. Descripción del Conjunto de Datos

El volumen de datos debe ser considerable por lo que obtuvimos todas las imágenes de los rostros de estudiantes con las que cuenta almacenadas la ESPOL. En total conseguimos un total de 15600 imágenes con un tamaño total de aproximadamente 800 MB. Esto nos dio la libertad de ejecutar pruebas variando el tamaño del conjunto de datos y obtener proyecciones más acertadas.

Gracias a la flexibilidad del SDK utilizado las imágenes no deben cumplir con una dimensión establecida. Por otra parte los formatos de las imágenes en su mayoría son JPEG, y minoritariamente PNG.

Cuando se trata de procesar multimedia muchas veces se deben hacer concesiones y balancear calidad de resultado con rendimiento, dependiendo de las necesidades que se tengan. En nuestro caso decidimos que para asegurar en cierta medida la calidad del resultado y no comprometer el rendimiento, los archivos deben ser de no menos de 512x512 píxeles.

9.4. Detalles de Implementación

Como se menciona en la sección 9.1 el flujo del procesamiento en Hadoop sigue el esquema básico de un trabajo MapReduce. No hace uso de fase intermedias y en la fase Reduce simplemente emite los resultados que recibe para que sean escritos en el archivo de salida.

En el siguiente fragmento de código mostramos como el valor recibido en el Mapper constituye todo el contenido de una imagen de nuestro conjunto de datos.

```
std::string strVar=context.getInputValue();
unsigned char* img;
img = (unsigned char*) strVar.c_str();
```

La imagen base que es sobre la cual se compara todo el conjunto de datos es obtenida de la memoria cache distribuida de Hadoop.

A partir de la recepción y representación en memoria de la imagen como una cadena de caracteres, toma el control de las imágenes la librería FaceSDK iniciando con la interpretación de la imagen.

```
HImage img1, img2;
FSDK_LoadImageFromJpegBuffer(&img1, buffer, lSize);
FSDK_LoadImageFromJpegBuffer(&img2,img, strVar.length)
```

El siguiente paso es detectar el rostro dentro de la imagen proporcionada al Mapper y la imagen base, y obtener la plantilla (template) del rostro. Pero justo antes de hacer esto, se proporcionan parámetros para la detección de rostros.

```
FSDK_SetFaceDetectionParameters(false, false, 512);
```

El primer parámetro le indica al SDK si debe o no manejar rotaciones arbitrarias del rostro; mientras que el segundo determina si la librería debe calcular el ángulo de rotación del rostro. El tercer y último parámetro le dice al SDK que tamaño utilizar para hacer el redimensionamiento interno de las imágenes antes de procesarlas, como se menciona en la sección anterior se requiere que las imágenes sean de no menos de 512 pixeles de ancho por 512 pixeles de alto.

Estos son los parámetros utilizados por defecto. En nuestro caso decidimos utilizar los mismos parámetros para tratar de requerir el menor procesamiento posible de imágenes, y así obtener un mejor rendimiento.

```
FSDK_FaceTemplate f1, f2;
FSDK_SetFaceDetectionThreshold(5);
FSDK_GetFaceTemplate(img1, &f1);
FSDK_GetFaceTemplate(img2, &f2);
```

En la última etapa estas plantillas obtenidas son utilizadas para comparar los dos rostros y así encontrar el grado de similitud entre ellos.

```
float similarity;
FSDK_MatchFaces(&f1, &f2, &similarity);
float thresh;
FSDK_GetMatchingThresholdAtFAR(0.5, &thresh);
```

Se obtiene un valor umbral (threshold) utilizado para filtrar los resultados obtenidos. El Mapper emite un resultado siempre y cuando el valor de similitud entre las plantillas sea mayor al umbral.

10. Pruebas y Resultados

Luego de realizar el proceso de adaptación de la librería de reconocimiento facial a nuestro programa de MapReduce, ejecutamos nuestra aplicación de procesamiento distribuido en la nube, es decir utilizando los Servicios Web de Amazon, realizando una serie de pruebas que se detallan a continuación:

10.1. Número fijo de Nodos

Hemos procesado diferente número de imágenes utilizando 14 nodos provistos por Amazon Web Services y obtuvimos los resultados como se muestra en la Figura 2

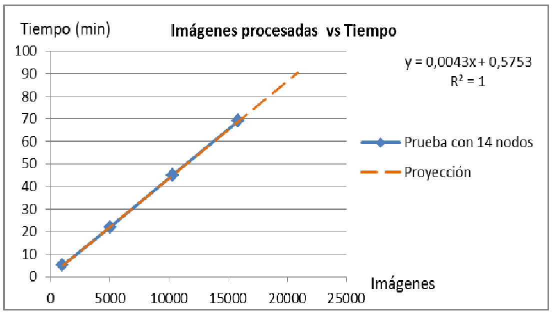

Figura 2: Prueba con 14 nodos

En la figura 2 se aprecia el comportamiento del tiempo que se toma al aumentar el número de imágenes en nuestra base de datos, se puede observar que estas dos variables tienen una relación lineal directamente proporcional.

10.2. Número fijo de Imágenes

Hemos procesado diferente número de imágenes variando el número de nodos provistos por Amazon Web Services fijando el número de imágenes en 1000 como se muestra en la Figura 3

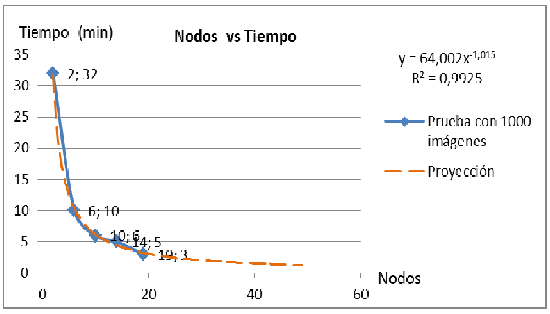

Figura 3: Prueba con 1000 imágenes

En la figura 3 se observa el comportamiento del tiempo que toma en procesar 1000 imágenes aumentando el número de nodos. Dado este grafico que representa una función de tipo potencial, podemos hacer predicciones acerca del tiempo de procesamiento si se tuvieran más nodos, este proceso de predicción se puede realizar aplicando métodos de regresión potencial con la ayuda de Excel.

Podemos notar que en el caso de las 1000 imágenes procesadas con diferente número de nodos, el coeficiente de determinación es 0.9925 con este resultado podemos decir que esta función potencial tiene una buena calidad de ajuste del modelo de regresión aplicado ya que el coeficiente de determinación se acerca a uno, por lo que la línea de tendencia calculada para este caso tiene una menor cantidad de error en la estimación.

11. Conclusiones

- El paradigma MapReduce, implementado en la plataforma Hadoop, es capaz de procesar gran cantidad de archivos de imágenes eficientemente.
- Es posible construir una solución de bajo costo para el procesamiento masivo de imágenes utilizando servicios del tipo Infraestructura como Servicio (IaaS por sus siglas en inglés) como el EC2 de Amazon.
- Pudimos constatar que con la ayuda de la plataforma de procesamiento distribuido, Hadoop, se logró procesar un número de imágenes mucho mayor de las que se pueden procesar utilizando solo un computador.
- La mejora en el rendimiento del procesamiento a medida que se incrementan los nodos es notable. El tiempo disminuye de manera exponencial con el aumento de nodos.
- La investigación acerca de la tecnología de reconocimiento facial nos dice que aún los algoritmos no son 100% precisos; y que dependen mucho de la calidad de las imágenes utilizadas. Es así que la mejor y utilización de esta tecnología dependerá mucho del avance que se de en las cámaras y demás equipos para capturar imágenes.

12. Recomendaciones

- A partir del presente trabajo podría construirse una interfaz gráfica para este módulo para poder proveer la imagen de entrada, así como también parámetros de configuración para él mismo.
- Se recomienda utilizar como entrada una imagen que contenga en su gran mayoría el rostro del individuo y que sea mayor a 512 x 512 pixeles. Si se utiliza imágenes de menor dimensión aumenta la probabilidad de obtener resultados poco precisos.

13. Referencias

[1] Wikipedia Contributors, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Face_detection&oldid=337481273, 2010 Jan

[2] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja, "Detecting Faces in Images: A Survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34-58, January 2002.

[3] Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in Conference on Computer Vision and Pattern Recognition (CVPR'01), vol. 1, Kauai, 2001.

[4] Henry A Rowley, Shumeet Baluja, and Takeo Kanade, "Human Face Detection in Visual Scenes," School of Computer Science, Carnegie Mellon University, Pittsburg, Paper 2005.

[5] Jeffrey Dean and Sanjay Ghemawat, Google, Inc., 2004.

[6] The Apache Software Foundation, Hadoop, [Online]. http://hadoop.apache.org/, 2009, Jul
[7] Yahoo, Hadoop and Distributed Computing at Yahoo, [Online]. http://developer.yahoo.com/hadoop/, 2010
[8] Yahoo, MapReduce, [Online]. http://developer.yahoo.com/hadoop/tutorial/module4.html#pipes, 2010