Procesamiento de video usando Apache Hadoop con OpenCV y JavaCV para reconocimiento facial

Video processing using Apache Hadoop with OpenCV and JavaCV to face recognition

Lucas Rogerio Garcés Guayta[a], Yasser Cesar Alvarado Salinas[b], Nixon Rafael Paladines Enríquez[c]

[a] Departamento de Eléctrica y Electrónica, Universidad de las Fuerzas Armadas ESPE Extensión Latacunga
[b] Facultad de Ingeniería en Sistemas, Telecomunicaciones y Electrónica, Universidad de Especialidades Espíritu Santo UEES
Maestrante en Auditoría de Tecnologías de la Información
[c] Escuela Técnica Superior de Ingeniería Informática, Universidad Rey Juan Carlos
Maestrante en Ingeniería de Sistemas de Información
Correspondencia Autores: lrgarces@espe.edu.ec, yalvarado@uees.edu.ec, nr.paladines@alumnos.urjc.es

Recibido: agosto 2016, **Publicado:** diciembre 2016

***Resumen*—** El presente artículo propone una solución informática de bajo coste, que permite realizar análisis de video para la identificación de personas haciendo uso de tecnología *open source* como Apache Hadoop y las librerías OpenCV y JavaCV. Estas herramientas permitirán el análisis, detección y reconocimiento facial, desde los videos alojados en una base de datos, alimentada no solo por las cámaras del ECU-911, sino también por todas las que se encuentran instaladas en establecimientos públicos y privados del país. La rapidez para procesar este gran volumen de información dependerá de los nodos implementados para ejecutar el MapReduce. Además, se indican los campos de aplicación para este sistema interconectado de video-vigilancia.

***Palabras Claves*—** Procesamiento de video, reconocimiento facial, Apache Hadoop, OpenCV, JavaCV

***Abstract*—** This paper proposes a low-cost computing solution, to perform video analysis to identification of people making use of open source technology such as Apache Hadoop and OpenCV and JavaCV libraries. These tools enable the analysis, detection and face recognition, from videos hosted on a database, fueled not only by the ECU-911 cameras, but also by all security cameras are installed in public and private establishments in Ecuador. The speed to process this huge volume of information depend on the implemented for running the MapReduce nodes. In addition, the fields of application for this interconnected system of video.

***Keywords*—** Video processing, face recognition, Apache Hadoop, OpenCV, JavaCV.

I. INTRODUCCIÓN

Actualmente la inseguridad ciudadana se ha convertido en un problema social, que afecta no solo al Ecuador sino también a otros países. Cada día se cometen actos delictivos en las calles, sea de día o de noche.

El trabajo que realiza el personal de la policía nacional para aprehender a los delincuentes es arduo, pero les toma mucho tiempo.

Por ello, implementó un sistema de recompensas a quien proporcionase información sobre los más buscados en el país.

El gobierno nacional, en su afán de combatir la delincuencia y minimizar la inseguridad, ha implementado en las principales ciudades del Ecuador cámaras de seguridad, denominadas ojos de águila.

Asimismo, en los medios de transporte como buses y taxis, se ha instalado kits de seguridad que incluye: un dispositivo de ubicación, dos cámaras y botones de auxilio; todo esto conectado al Servicio Integrado de Seguridad ECU-911. Los videos captados por las cámaras les han permitido a la policía nacional actuar de forma rápida y eficiente, logrando identificar, detener y sentenciar a muchos, pero no a todos.

En este sentido, aprovechando que la detección y reconocimiento facial ha sido investigada ampliamente, y que gracias a ello se han construido sistemas informáticos de alto rendimiento, capaces de reconocer un rostro en entornos distintos, como por ejemplo: aeropuertos, centros comerciales, estadios, etc.; se presenta una propuesta como solución informática, con la capacidad de realizar análisis de video para el análisis, detección y reconocimiento de personas, haciendo uso de tecnología libre.

En este artículo se detallan las herramientas a utilizar para el manejo y análisis de grandes volúmenes de información, así como la tecnología utilizada para el reconocimiento facial. También se hace una descripción general de la integración de los componentes a implementar. Al final se analizan los resultados y también se presentan los posibles campos de aplicación.

II. GENERALIDADES

Antes de iniciar es necesario definir y diferenciar los términos detección facial y reconocimiento facial. Luego se describirán los componentes principales y se detallará la propuesta.

A. Detección y reconocimiento facial

Son términos diferentes, pero están muy relacionados. Para hacer reconocimiento facial es necesaria una detección facial.

El proceso de detección facial [1] hace uso de algoritmos para determinar si hay rostros humanos en una foto o video, es decir, solo determina si hay alguna cara, mas no a quien pertenece.

En cambio, el reconocimiento facial [2] identifica el rostro de una persona en una foto o video, comparándolas con las imágenes que ya se encuentran previamente almacenadas en la base de datos.

Estas imágenes del rostro deben ser registradas siguiendo ciertas reglas de calidad, como la iluminación, frontalidad o el tamaño de la cara en píxeles.

B. Adquisición de información a escalas de Big Data

Big Data [3], [4] ya no es una palabra desconocida, cada vez se hace más evidente y necesario el uso de este término para referirse a grandes conjuntos de información, incluyendo datos semiestructurados y no estructurados que resultan difíciles de administrar y analizar con herramientas tradicionales [5].

Las fuentes de este flujo de datos son de dos tipos: huellas digitales generadas por los humanos y los datos de máquina [6]. Las primeras pueden ser un clic, comentarios de foros o redes sociales, correo electrónico, entre otros. Los datos de máquina corresponden a los sensores, radares, satélites, cámaras de video, servidores, entre otros.

Ahora bien, en este caso, los datos que serán procesados y analizados podrían provenir de los video captados por las cámaras de seguridad del ECU-911. Según informes publicados en el portal web de esta institución, indica que tienen instaladas más de 2000 cámaras, distribuidas en diferentes sectores de las principales ciudades del Ecuador.

En efecto, se instalaron 55000 kits de seguridad en los transportes, de los cuales 17000 pertenecen a buses interprovinciales y 38000 a unidades de taxis como se observa en la figura 1.

Según la Agencia Nacional de Tránsito aún faltan de instalar cámaras en 14500 unidades de taxis. Estas cámaras instaladas deben detectar los rostros de todas las personas que hacen uso de estos medios de transporte.

Además, existen otras fuentes de información que deben ser consideras por la importancia que representan, estas son las cámaras de seguridad de establecimientos públicos y privados.

Principalmente, deben estar interconectados los lugares de mayor concurrencia de personas, como, por ejemplo: restaurantes, bares, hoteles, aeropuertos, cajeros automáticos, mercados y centros comerciales.

Por lo tanto, todas las cámaras de seguridad interconectadas, proporcionarán diariamente imágenes en tiempo real de todas las personas capturadas en video. Para lograr almacenar, gestionar y manipular este gran volumen de información será necesario utilizar una base de datos NoSQL (*not only SQL*, no solo SQL), además una plataforma de análisis como el Framework Apache Hadoop y para la detección y reconocimiento facial se hará uso de las librerías de OpenCV, FFmpeg y JavaCV.

C. Framework Apache Hadoop

Apache Hadoop [7] es un software libre que soporta aplicaciones distribuidas para almacenar, procesar y analizar grandes volúmenes de datos estructurados y no estructurados, sean estos cientos de terabytes, petabytes, zettabyte o yottabyte. Su diseño permite a las aplicaciones trabajar con miles de nodos. Ofrece un sistema robusto tolerante a fallos puesto que usa una arquitectura *Master-Slave*, para almacenar su sistema de archivos distribuidos Hadoop (*Hadoop Distributed File System*, HDFS), inspirado en el sistema de archivos de Google y el algoritmo de *MapReduce*, para hacer cálculos mediante el procesamiento paralelo a través de los nodos del clúster.

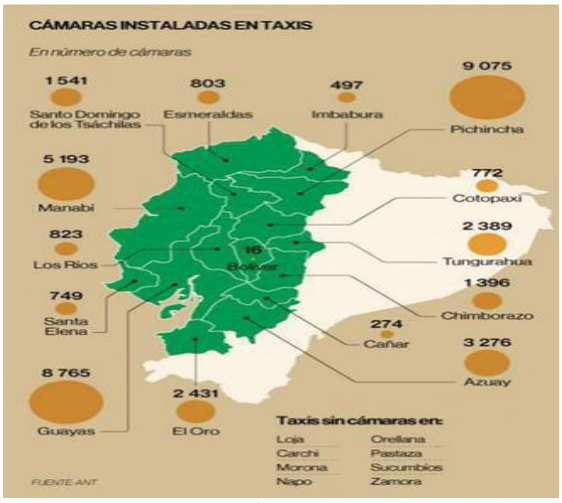

Figura 1. Total de cámaras instaladas en taxis
Fuente: Agencia Nacional de Tránsito, 2016 [8]

Su sistema de archivos distribuidos facilita la rápida transferencia de datos entre nodos y permite que el sistema siga funcionando sin interrupción en caso de un fallo. Además, el uso del código [9] escrito en otros lenguajes como Python y C es posible a través de *Hadoop Streaming*. Trae incorporado Hadoop Job y Trackers, que realizan el seguimiento de la ejecución de los programas a través de los nodos del clúster.

D. Arquitectura principal de Apache Hadoop
1. MapReduce

El algoritmo MapReduce [10 pág. 177] es un modelo de computación distribuida basado en Java, permite escribir aplicaciones que procesan grandes cantidades de datos estructurados y no estructurados en paralelo a través de un grupo de miles de máquinas, de una manera fiable y tolerante a fallos.

Contiene dos operaciones importantes: el *Map* (*mapper*) y el *Reduce*. La primera se encarga del mapeo y se aplica a los datos de entrada, donde toma un conjunto de datos y lo convierte en otro conjunto de datos agrupándolos en una lista ordenada en pares clave/valor [11 pág. 2]. En cambio la tarea *Reduce* toma la salida del *Map* como entrada, combinando toda la lista de pares ordenados y agrupa los valores por medio de las claves.

Ayuda a los desarrolladores [12] a realizar estos dos pasos de manera eficiente. Por lo tanto, Hadoop hace que sea fácil de procesar grandes conjuntos de datos al permitir que los desarrolladores se centren en la lógica de la base datos en vez de preocuparse por la complejidad y el tamaño de los datos.

2. HDFS

El Sistema de Archivos Distribuido Hadoop (HDFS) está diseñado para almacenar grandes conjuntos de datos de forma fiable, y para transmitir los conjuntos de datos con alto ancho de banda para aplicaciones de usuario.

Tiene una arquitectura master/esclavo. Un clúster HDFS consta de una sola *NameNode*, un servidor maestro que gestiona el espacio de nombres del sistema de archivos y regula el acceso a los archivos de los clientes. Además, hay una serie de *DataNodes*, por lo general un nodo por clúster, que gestionan de almacenamiento conectado a los nodos que se ejecutan.

HDFS expone un espacio de nombres del sistema de archivos y permite que los datos de usuario se almacenen en archivos. Internamente, un archivo se divide en uno o más bloques y estos bloques se almacenan en un conjunto de *DataNodes*.

El *NameNod*e ejecuta operaciones de espacio de nombres del sistema de archivos como abrir, cerrar, y renombrar archivos y directorios. También determina la asignación de bloques para *DataNodes*. Los *DataNodes* son responsables de atender las solicitudes de lectura y escritura de los clientes del sistema de archivos, realizan la creación de bloques, eliminación, y la replicación en la instrucción de la NameNode.

El *NameNode* y *DataNode* [13] son piezas de software diseñados para funcionar con sistema operativo GNU / Linux. HDFS utiliza Java, cualquier máquina que soporte este lenguaje puede ejecutar el *NameNode* o el *DataNode*.

3. Hive

Apache Hive [14 pág. 28] fue construido inicialmente por Facebook como solución de almacenamiento de datos de código abierto construido en la cima de Hadoop para facilitar el acceso a MapReduce. Permite la gestión de grandes conjuntos de datos que residen en almacenamiento distribuido.

Hive [15] proporciona un lenguaje similar a SQL conocido como *HiveQL* (*Hive Query Language*). Estas consultas se compilan automáticamente en trabajos *MapReduce* que se ejecutan usando Hadoop. *HiveQL* permite a los usuarios convertir scripts MapReduce personalizados en consultas.

Admite instrucciones del lenguaje de definición de datos (DDL), que usan para crear, eliminar y alterar tablas de una base de datos. Permite cargar datos desde fuentes externas e insertar los resultados de las consultas en tablas Hive, a través de las instrucciones del lenguaje de manipulación de datos (DML).

Sin embargo, tiene restricciones al momento de actualizar y borrar filas de las tablas existentes. El componente MetaStore es el catálogo del sistema Hive, que almacena metadatos sobre la tabla subyacente como sus columnas, privilegios, y más. Estos metadatos se especifican durante la creación de la tabla y se reutilizan cada vez que se referencia la tabla en HiveQL.

4. Fuse-DFS Project

Fuse DFS [16] es un subproyecto de Apache Hadoop, proporciona una interfaz para salvar la brecha entre el HDFS y el sistema de archivos local, de manera que miles de bibliotecas de programación diseñadas para el sistema de archivos local pueden tomar la ventaja de Hadoop. Admite muchas operaciones como leer y escribir, y las operaciones de directorio; sin embargo, no es totalmente compatible con POSIX.

5. OpenCV, FFmpeg y JavaCV

OpenCV y *FFmpeg* son librerías que se utilizan frecuentemente en visión por computador y proyectos de procesamiento de video y reconocimiento facial. Sin ellos el procesamiento seguiría siendo teoría en lugar de implementaciones. Son diseñadas y desarrolladas para C++ mientras que Hadoop se ha desarrollado y diseñado para Java.

Para solucionar esto se ha creado un proyecto denominado JavaCV, que es una librería de código abierto que proporciona una solución estable y eficiente para hacer uso de los beneficios OpenCV y FFmpeg, así como muchos otros procesamientos de video en múltiples plataformas como Windows, Linux, Mac OS y Android, con la aceleración de hardware compatible [17].

FFmpeg [18], es una biblioteca de procesamiento de video que ofrece la fluidificación de grabaciones de audio y de video, conversión y una solución completa. Contiene una biblioteca de códec de audio/video muy avanzada que garantiza la calidad de alta portabilidad. Es muy utilizada para analizar el video, fotogramas y enviar datos *OpenCV* para su procesamiento.

E. Descripción general de la integración del sistema

Con los componentes previamente detallados se puede empezar a describir la integración de Apache Hadoop con las librerías OpenCV y JavaCV, para la construcción del

sistema de procesamiento de video para el reconocimiento facial. En la estructura del sistema se destacan 4 aspectos importantes que son: 1) El HDFS que ofrece el servicio de almacenamiento distribuido para datos del video; 2) Fuse DFS se carga el sistema de archivos distribuido en el sistema de archivos local; 3) JavaCV mediante dos puertos llama a las librerías de procesamiento de video: OpenCV y FFMPEG, con lenguaje Java para que estén disponibles para todo el procesamiento de video; 4) El modelo de programación *MapReduce* de Hadoop procesa datos de video al mismo tiempo [17].

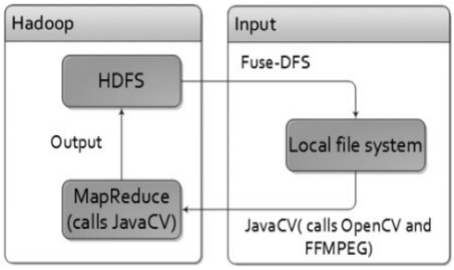

Figura 2. Descripción del sistema
Fuente: Tan H y Chen L., 2015 [17]

La idea clave del modelo de programación *MapReduce* es encapsular los datos en pares de clave-valor, de modo que los *Mappers* y *Reducers* puede procesar simultáneamente en líneas de corriente paralelas, lo que aumenta drásticamente el rendimiento del sistema.

Se puede observar en la figura 2 que Fuse DFS carga el HDFS al sistema de archivos local y hace que los datos de video almacenados en HDFS estén disponibles para JavaCV. JavaCV, que hereda poderosa capacidad de análisis de video de OpenCV y FFMPEG, hace que las bibliotecas de video estén a disposición de todo el procedimiento para el *MapReduce*.

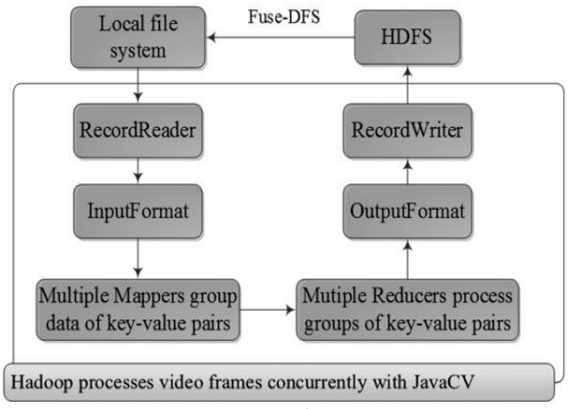

Figura 3. Modelo de programación del sistema Hadoop propuesto
Fuente: Tan H y Chen L., 2015 [17]

Se corta el video usando FFmpeg, dividiéndolo en varios clips y cada uno de ellos se procesa como una unidad Hadoop. El usuario puede extraer el resumen de videos mediante fotograma clave y entender brevemente su contenido. FFmpeg podría utilizarse para analizar cualquier formato de video en la secuencia que podría ser procesada por OpenCV.

El programa de procesamiento de imágenes OpenCV se puede modificar arbitrariamente para completar diferentes funciones [18 pág. 2187].

El procedimiento de video [17] [19] puede describirse a continuación: 1) *RecordReade*r lee los datos de video de HDFS a través de la interfaz proporcionada por JavaCV, que encapsula los datos en pares clave-valor (Identificación del marco, datos del marco) y los somete a *InputFormat*; 2) Una *InputFormat* puede aceptar varios tipos de pares clave-valor proporcionados por *RecordReaders* e *InputFormat* combina todos los pares clave-valor para someterlos a *mappers*; 3) El grupo de *mappers* de pares clave-valor de acuerdo a los requerimientos de los algoritmos los envían a *Reducers*; 4) Cada *Reducer* procesa un grupo de pares clave-valor y presenta los resultados a *OutputFormat*; 5) *OutputFormat* emplea *RecordWriter* para escribir resultados al HDFS [20 pág. 4080].

1. Entrada de video

Antes que nada se requiere leer los fotogramas del video para luego aplicar los algoritmos correspondientes. El primer paso es cargar fotogramas de los archivos de video que se almacenan en HDFS con JavaCV. El segundo paso es transformar el tipo de datos de los marcos de JavaCV (IplImage) en pares clave-valor de Hadoop (<texto, BytesWritable>), donde la clave es un identificador único y el valor es el cuadro de imagen correspondiente en bytes. Al llegar a la programación, estos dos pasos se encapsulan en una clase Java *VideoRecordReader* que se extiende desde la clase abstracta *RecordReader* de Hadoop [21].

2. Estrategia del Map

El objetivo principal del *Map* es pasar los grupos de fotogramas de video al *Reducer*. Por lo general, hay dos tipos de aplicaciones de procesamiento de video.

El primero, el análisis orientado a un solo cuadro. Un ejemplo de esta aplicación es la detección de rostros. Este es un trabajo repetido en cada fotograma sin correlación entre cuadros. Simplemente se pasa pares clave-valor (Identificación del marco, los datos de trama) para reducir el proceso.

El segundo, es el *frame-series* orientado al análisis de video. Registrando detección de movimiento y de seguimiento, para ello se requiere una serie de fotogramas en orden de tiempo para el análisis. Para este tipo de aplicación se combina pares clave-valor de <Identificación del *frame*, datos del *frame*> en pares clave-valor de <Id serie del *frame*, los datos de serie del *frame*>. Cada serie del *frame*, constituye un subproblema similar del problema original y se puede ejecutar simultáneamente por los *Reducer* de Hadoop. Lo único nuevo es asegurar que es posible sintetizar resultados de los subproblemas para obtener resultados del problema original. Para la detección

de movimiento y de seguimiento las series de *frames* se supone que son parcialmente solapadas, por lo que se puede vincular los resultados de diferentes series de acuerdo a los marcos superpuestos.

3. Estrategia del Reducer

La tarea del *Reducer,* como muestra la figura 3, es llevar a cabo las operaciones principales de procesamiento de video para la detección de imagen. Los datos de entrada para reducir son una serie de fotogramas de video. Para el análisis de video orientado a un solo cuadro, las tareas *Reducer* simplemente realizan algoritmos como en un solo equipo.

Con la ayuda de JavaCV, la implementación no es difícil. Para los *frame-serie* orientado al análisis de video, las tareas *Reducer* toman la serie del marco de entrada como un pequeño periodo de video y también realizan algoritmos como en un solo equipo [17] [19].

4. Proceso posterior

Los resultados de la tarea del *Reducer* del *frame-serie* de análisis de video orientado, pueden requiere algún proceso posterior para formar un resultado monolítico. De acuerdo con los números de orden de los archivos de salida y marcos superpuestos de los contenidos de video, un resultado monolítico puede ser construido.

5. Detección facial

En caso de querer localizar todos los rostros en cada cuadro de los archivos de video de entrada y generar registros de texto, como se indicó anteriormente, este es un trabajo repetido en cada fotograma sin correlación entre cuadros. Así que el orden del grupo de Map no importa. En el *Reducer*, cada cuadro se analiza con un clasificador de detección de rostro humano, que es una cascada de clasificadores de características similares llamadas *Haar-Like* implementados con JavaCV [2], [22]. Los resultados de detección son lugares de rostros en cada cuadro y escrito a HDFS.

F. Detección de movimiento y seguimiento

Esto aplica si se desea detectar todos los objetos que se mueven en los archivos de video de entrada y realizar un seguimiento de ellos. Los registros se generan como resultado, cada registro incluye el número de fotograma de su cuadro de inicio, *frame* final y una serie de rectángulos que indican la ubicación de los objetos en movimiento en los fotogramas intermedios. En este caso, los *frame* deben ser agrupados por orden de número de fotograma o tiempo. Así que estos grupos *frame* son series del *frame*. Cada serie puede ser vista como una parte de video y se envía al *Reducer*.

La detección de movimiento basado en estadística modelo de fondo y seguimiento de objetos por el filtro de Kalman, se implementa en el *Reducer* para obtener resultados de cada serie del *frame*. Estas *frame-series* son parcialmente solapadas, por lo que se puede combinar resultados de diferentes series de acuerdo a los marcos superpuestos [17].

G. Campos de aplicación

El FBI [23] implementó un sistema de identificación denominado *Next Generation Identification System,* es un gran buscador humano que cuenta con una base de datos con más de 51 millones de fotografías, realiza 196 búsquedas y reconocimientos al día; reduciendo el tiempo de identificación de personas de 2 horas a 10 minutos y la espera de verificación de antecedentes disminuyó de 24 horas a 15 minutos. El sistema también registra otros datos, como son las cicatrices, tatuajes y marcas de nacimiento.

Aunque el reconocimiento facial está relacionado con los temas de seguridad, existen otros campos donde es muy útil, como, por ejemplo: verificar la asistencia de los alumnos en un salón de clase o de los trabajadores en una determinada empresa, constatar la identidad de la persona sin necesidad de presentar la cédula, reconocer a los pasajeros en un aeropuerto, identificar a los clientes en un centro comercial y conocer sus preferencias.

III. RESULTADOS

Al verificar el rendimiento de la propuesta, la ejecución a pequeña escala del sistema es muy satisfactoria, resaltando que la rapidez dependerá de los nodos esclavos para ejecutar las tareas del *Map Reduce*, tal como lo demuestran Tan Hanlin y Chen Lidong [17] en su prueba experimental donde utilizaron seis equipos para realizar el experimento, de los cuales uno actuó como master (*Namenode y JobTrack*er) y el resto como esclavos (*Datanode y TaskTracker*).

En el experimento se considera el tiempo como variable de observación y el número de archivos de video de entrada estándar junto al número de esclavos en el clúster como variables de control. Un archivo de video de entrada estándar contiene 690 *frames* (15fps x 46 segundos) con una resolución de 320 x 240 píxeles para cada cuadro. El tamaño del archivo es de 2,6 MB.

En cuanto a los resultados experimentales en la detección facial, la figura 4 y 5 muestran el rendimiento utilizando Hadoop, de estos resultados se puede concluir que el tiempo de funcionamiento: 1) aumenta linealmente con aproximadamente el tamaño de archivos de video de entrada, lo que es natural desde la detección de la cara que es un simple trabajo repetido en los clústers; 2) Disminuye considerablemente con el aumento del número de esclavos, cuanto más tiempo la tarea consume en un solo equipo, es

mayor el tiempo que ahorra para realizar la tarea con Hadoop cluster. Un clúster con cinco esclavos puede reducir el tiempo de ejecución a menos del 25% de la de un solo equipo. Y se puede deducir que el tiempo de funcionamiento se puede reducir con más nodos añadidos [17].

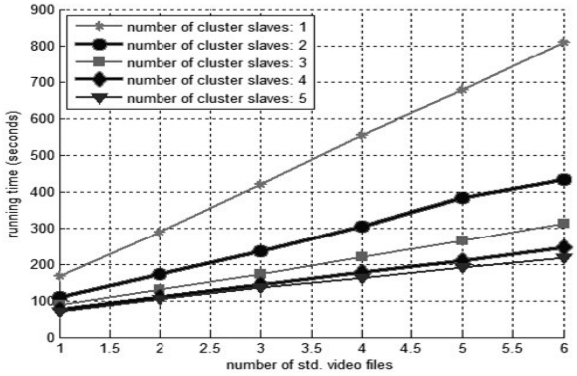

Figura 4. Tiempo de funcionamiento con un número diferente de videos estándar de entrada
Fuente: Tan H y Chen L., 2015 [17]

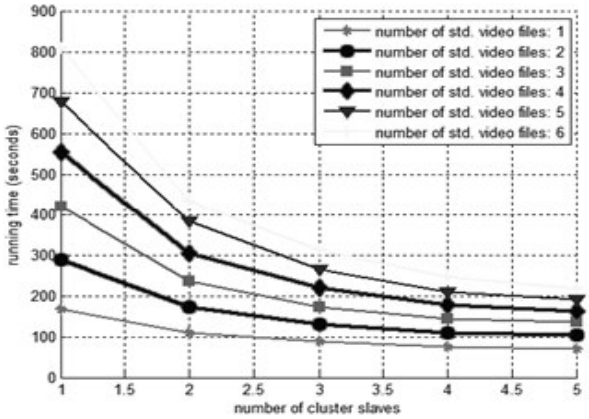

Figura 5. Tiempo de funcionamiento con un número diferente de esclavos
Fuente: Tan H y Chen L., 2015 [17]

Con el mismo sistema propuesto, además de hacer reconocimiento facial de videos también se lo puede hacer utilizando imágenes, que reemplazarían a los fotogramas y el rendimiento sería mayor.

En la figura 6 se muestra la relación inversamente proporcional que existe entre los nodos y el tiempo. Entre más nodos se utilicen menor será el tiempo de respuesta.

IV. DISCUSIÓN

La preocupación principal de las pruebas es el rendimiento del sistema, puesto que, para mejorar la detección facial o la detección de movimiento y seguimiento, se necesita implementar algoritmos básicos de detección facial a través de características como la resta de *frames* y filtro de Kalman utilizando la interfaz de JavaCV (Java de OpenCV *Haar-like*).

Figura 6. Tiempo vs número de nodos
Fuente: Merchan A, Plaza J y Moreno J, 2010 [19]

En la figura 7 se muestra los resultados de procesar 15857 imágenes en 69 minutos utilizando 14 nodos [19].

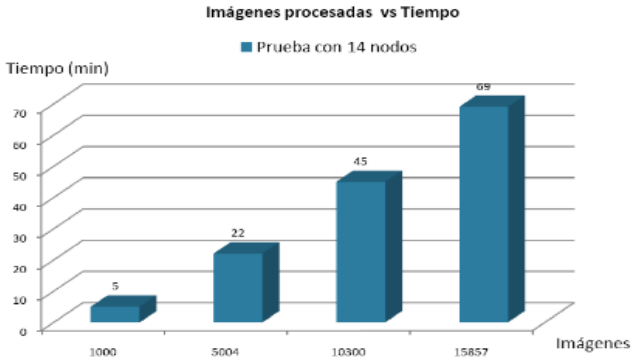

Figura 7. Imágenes procesadas vs tiempo
Fuente: Merchán A, Plaza J y Moreno J, 2010 [19]

V. CONCLUSIONES

La propuesta con *Apache Hadoop*, *Map Reducer* y OpenCV para el reconocimiento facial, es muy eficiente pero no es la única, existen algunas que utilizan algoritmos de aprendizaje o redes neuronales para lograr este fin, como por ejemplo: red neuronal *Kohonen*, algoritmo de aprendizaje de Adaboost [2], análisis de los componentes (PCA), análisis discriminante lineal (LDA), patrón binario local (LBP), patrón ternario local (LTP), entre otros [24]. Se pueden integrar algunos de estos algoritmos a la solución expuesta, para incrementar su eficiencia en tiempo real.

Además, como se indicó en la introducción, la delincuencia cada vez incrementa su accionar, por ello sería recomendable implementar esta propuesta para identificar a personas que son buscadas por la justicia. Técnicamente sería muy factible, ya que los equipos necesarios para capturar los videos se encuentran instalados, solo faltaría hacer el procesamiento de video para la detección y reconocimiento facial. De esta forma, si una persona que es buscada por las autoridades hace uso de un taxi, se sube en un bus, acude a un cajero automático, entra en un centro comercial o que sea captado por alguna de las cámaras de seguridad interconectadas; será reconocido al instante.

En caso de implementarse será de vital importancia conectarse con la base de datos de las fotografías del registro civil, para obtener los datos personales de quien se haya reconocido facialmente. Esta implementación contribuirá a la video-vigilancia de una ciudad inteligente.

Talvez la violación a la privacidad de los ciudadanos pueda ser un impedimento para su implementación, por lo que se considera que se deben incorporar o modificar las políticas existentes, permitiendo el uso de este tipo de herramientas.

VI. REFERENCIAS

[1] Sistema de reconocimiento facial basado en imágenes con color. **PEDRAZA PICO, BEATRIZ OMAIRA, RONDÓN, PAOLA y ARGVELLO, HENRY.** 2, Colombia : UIS Ingenierias, 2011, Vol. 10.

[2] Reconocimiento de Facial basado en FPGA. **Molina, Julio C. y Risco, Miguel A. .** 1, Perú : Revista ECIPerú, 2011, Vol. 8. ISSN 1813-0194.

[3] **Intel IT Center.** Dialogo TI Intel. [En línea] Junio de 2014. [Citado el: 08 de Julio de 2015.] http://dialogoti.intel.com/sites/default/files/documents/ e7_big_data_planning-guide_v2d_esp.pdf.

[4] Big data. Un nuevo paradigma de análisis de datos. **Jiménez, Carlos Maté.** 2, España : anales, 2014, Vol. 1.

[5] An Architecture for Big Data Analytics. **Chan, Joseph O.** USA : Communications of the IIMA, 2013, Vol. 13.

[6] Framework para la gestión, el almacenamiento y la preparacion de grandes volúmenes de datos. Big Data. **Almeida Pazmiño, Marco Antonio, Lara Torralbo, Juan Alfonso y Lizcano Casas, David.** 1, España : CEF, 2015.

[7] MapReduce programming with apache Hadoop. **Bhandarkar, M.** Atlanta : IEEE, 2010. 978-1-4244-6442-5.

[8] **ANT, Agencia Nacional de Tránsito.** www.ant.gob.ec. Agencia Nacional de Tránsito. [En línea] [Citado el: 10 de Junio de 2016.] www.ant.gob.ec.

[9] An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. **Taylor, Ronald C.** 11, EE.UU. : BMC Bioinformatics, 2010. 1471-2105-11-S12-S1.

[10] Procesamiento de datos de texto-intensivo con MapReduce. **Lin, Jimmy y Dyer, Chris .** 1, 2010.

[11] Efficient Parallel Set-Similarity Joins Using MapReduce. **Rares, Vernica, Carey, Michael J. y Li, Chen .** EEUU : SIGMOD, 2010. 978-1-4503-0032.

[12] **MSV, Janakiram.** Your Story. [En línea] 8 de Julio de 2012. [Citado el: 10 de Julio de 2015.] http://yourstory.com/2012/07/what-is-common-between-mumbai-dabbawalas-and-apache-hadoop/.

[13] The Hadoop Distributed File System. **Shvachko, Konstantin , y otros.** 1, California USA : IEEE, 2010. 978-1-4244-7153-9.

[14] Comparison of SQL with HiveQL. **Kumar, Rakesh , y otros.** Jaipur, India : IJRTS, 2014, Vol. 1. 2348-1439.

[15] **Sakr, Sherif.** IMB. [En línea] 14 de Agosto de 2012. [Citado el: 10 de Julio de 2015.] http://www.ibm.com/developerworks/ssa/opensource/library/ os-mapreducesql/

[16] HDFS Space Consolidation. **Mehta, Aastha, y otros.** 2, India : Birla Institute of Technology and Science, 2011.

[17] AN APPROACH FOR FAST AND PARALLEL VIDEO PROCESSING ON APACHE HADOOP CLUSTERS. **Tan, Hanlin y Chen, Lidong .** 1, China : s.n., 2015.

[18] A Kind of Video Abstracting System Base on Hadoop. **Hongyi, Li, y otros.** 2186-2191, Switzerland : Trans Tech Publications, 2014, Vols. 687-691.

[19] **Merchan, Ángel, Plaza, Juan y Moreno, Juan .** 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). Guayaquil : CENTRO DE INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA (ESPOL), 2010.

[20] Dynamic Resource Allocation And Distributed Video Transcoding Using doop Cloud Computing. **Shanthi, B.R y Narayanan.C, Prakash .** India : JIRCCE, 2014, Vol. 2. 2320-9801.

[21] Face Detection System for Attendance of Class' Students. **Fuzail, uhammad, y otros.** 4, Pakistan : INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING, 2014, Vol. 5. 2045-7057.

[22] **Bracamonte Nole, Teresa J. y Huamán, Liz S. R. Pedro.** Aplicación de a Red Neuronal Kohonen al Reconocimiento de Rostros. Trujillo, Perú : Universidad Nacional de Trujillo , 2010.

[23] **Pagliery, Jose y CNNMoney.** CNN Español. [En línea] 17 de Septiembre de 2014. [Citado el: 09 de Julio de 2015.] http://cnnespanol.cnn.com/2014/09/17/el-fbi-lanza-un-poderoso-sistema-de-reconocimiento-facial/.

[24] Local Directional Pattern (LDP) for Face Recognition. **Jabid, Taskeed, Kabir, Hasanul y Chae, Oksam .** Corea del Sur : IEEE, 2010, Vol. 10. 78-1-4244-4 316-1.