Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

Arquitectura *Big Data* de colas para el procesado de vídeo en tiempo real

Presentado por José Luis Garrido Labrador
en Universidad de Burgos — 5 de mayo de 2020
Tutor: Dr. Álvar Arnaiz González y Dr. José Francisco Díez Pastor

Universidades de Burgos, León y Valladolid

Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

Dr. D. Álvar Arnaiz González, profesor del departamento de Ingeniería Informática.

Expone:

Que el alumno D. José Luis Garrido Labrador, con DNI 71707244Y, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado Arquitectura *Big Data* de colas para el procesado de vídeo en tiempo real.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 5 de mayo de 2020

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
Dr. D. Álvar Arnaiz González	Dr. D. José Francisco Díez Pastor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Memoria

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Los objetivos del proyecto se han dividido en tres apartados siendo estos los objetivos generales, los técnicos y los personales.

2.1. Objetivos generales

- Exploración de las diferentes herramientas para el procesado de vídeo en tiempo real a través de las fases de emisión, recogida, encolado, ingestión, procesado, enriquecimiento y almacenamiento.
- Estudio del estado del arte en análisis de imagen para el diagnóstico y tratamiento de enfermedades ante distintos escenarios tanto en aspectos físicos (iluminación, enfoque...) como en aspectos lógicos (resolución, tasa de refresco...).
- Implementación del software necesario para la recogida de vídeo en tiempo real sobre sistemas de videoconferencia.

2.2. Objetivos técnicos

- Crear una infraestructura software basada en contenedores *Docker* para ser independientes del software anfitrión y facilitar su despliegue.
- Desplegar un *pipeline* sobre herramientas de la suite de *Apache* para el *Big Data* que satisfagan el flujo ETL propuesto en la Figura 2.1.
- Desarrollar algoritmo sobre *Spark Stream* que procese los vídeos generando los datos necesarios para los estudios posteriores.

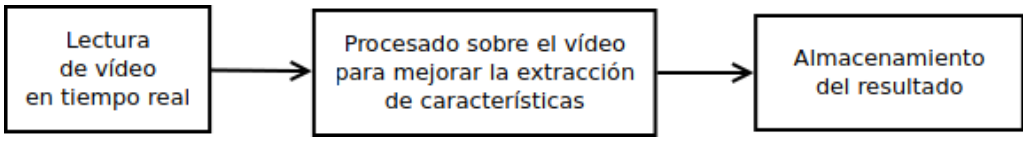

Figura 2.1: Flujo ETL objetivo del proyecto

2.3. Objetivos personales

- Contribuir a la mejora de la calidad de vida, a través de facilitar soportes para la telerehabilitación, de pacientes con enfermedad de Parkinson.
- Conocer más profundamente las herramientas de la suite de *Apache* y como estas se pueden combinar para facilitar tareas de *Big Data*.
- Completar mi formación durante el máster a través de la creación de una solución que utiliza gran parte de los conocimientos adquiridos durante el mismo.

Conceptos teóricos

Técnicas y herramientas

4.1. Gestión de flujo

Uno de los puntos esenciales de este trabajo es recoger y dirigir los *streams* de vídeo que se reciben. Por tanto, escoger una correcta aplicación para la gestión de este flujo de datos es una parte muy importante dentro de las herramientas.

Dentro de la suite de *Apache* existen varios componentes que se encargan de la gestión del flujos de datos. Se necesitan dos herramientas principales, una capaz de dirigir el flujo y otra para procesarlo.

Herramientas para dirigir el flujo

La primera herramienta necesaria debe ser capaz de dirigir flujos de datos, concretamente flujos de datos serializados para soportar datos más diversos como son las imágenes. También se necesita que sea capaz de desplegar diferentes colas para discriminar fácilmente a los distintos flujos de vídeo entrante.

En la suite de *Apache* existen dos herramientas que gestionan flujos de datos, son ***Apache Flume*** [1] y ***Apache Kafka*** [3].

Apache Flume es herramienta de gestión de flujo diseñada para hacer una gestión distribuida de manera fiable y altamente disponible de los datos. Proporciona un servicio eficiente para la recogida, agregación y almacenamiento de los datos. Sin embargo, aunque esta herramienta pudiese suplir las necesidades de una gestión de flujo se ha descartado debido a que está optimizado para la gestión de *logs*, concretamente datos codificados como cadenas, y no tiene un sistema de colas.

Apache Kafka es un proyecto de intermediación de mensajes que trabaja sobre el patrón publicación-suscripción funcionando como un sistema de transacciones distribuidas. Incorpora para la implementación de este patrón un sistema de colas para la distribución de mensajes. Aporta una API para el productor, el consumidor, el flujo y el conector y la conexión se realiza a través del protocolo de la capa de transporte *TCP*. Se va a utilizar esta herramienta al aportar un sistema de colas distribuidas fiable y soportar datos serializados.

Herramientas para procesar el flujo

En segundo lugar se necesita una herramienta capaz de procesar flujos de datos de forma escalable. La suite de *Apache* tiene dos herramientas principales para el procesado de información, son ***Apache Hadoop*** y ***Apache Spark***, ambas con extensiones para el procesado de flujos.

Apache Spark Streaming [4] aporta una *API* de consumidor nativo de *Kafka*, *Flume*, los sistemas de ficheros *HDFS* y *S3* entre otras herramientas. El funcionamiento interno consiste en crear pequeños lotes de datos para pasarlo al motor de *Spark* y retornar los lotes procesados.

Apache Hadoop Streaming [2] tiene un funcionamiento similar a *Spark Streaming* pero sin aportar de manera nativa una integración con *Kafka* y *Flume*.

Se ha escogido *Spark Streaming* frente a *Hadoop Streaming* por varios motivos:

1. *Spark Streaming* tiene integración con *Kafka* de manera nativa.
2. *Spark Streaming* hace un uso más intensivo de la memoria RAM, por lo que es mucho más rápido si se cuenta con una gran cantidad de esta[1].

4.2. Infraestructura de bajo nivel

Otro apartado importante en el despliegue de la aplicación son las herramientas y técnicas a ser usadas para la producción. Para esto se utilizan:

[1] El programa final se ha desplegado sobre una máquina con 128 GB de RAM y se ha probado en una de 32 GB de RAM con buenos resultados.

- ***GNU/Linux***, el sistema operativo más extendido en el entorno de los servidores [6, 8] además de estar disponible en los servidores prestados para la realización de este proyecto[2].

- ***Docker***, un software de gestión de contenedores estandarizados, semejante a los entornos *chroot* que facilita la virtualización de software en un entorno seguro y ligero. Sobre este motor se ejecutarán las aplicaciones del entorno de *Apache Spark* [5], *Apache Kafka* [7] además de la aplicación desarrollada para el cumplimiento de los objetivos.

[2]Se utilizan el servidor *Alpha* del GIR ADMIRABLE para el despliegue de la herramienta de recolección de datos (Procesador *Intel Core i7*-8700, 6 núcleos, 3.2GHz. 64GB de memoria RAM. 2 GPUs GTX 1080Ti y 500GB de disco duro sólido y 6 TB de disco duro magnético) y el servidor *Gamma* del mismo grupo para el despliegue de las colas y el procesado de la información (TODO Datos de gamma)

Aspectos relevantes del desarrollo del proyecto

En este capítulo se explicarán las partes más importantes del desarrollo del proyecto.

5.1. Desarrollo de la aplicación web

Para realizar la primera fase del desarrollo, la recogida de los datos, se ha desarrollado una aplicación web que facilitase las tareas de telerehabilitación conectando al personal especializado con los pacientes con enfermedad de parkinson y que dentro de sus funciones estuviese grabar a los mismos durante la realización de los ejercicios para ser estudiados en tiempo real.

Esta aplicación se ha compuesto de dos partes. Una ha consistido en el diseño y consecuente implementación de una interfaz hombre-máquina fácil de usar y accesible mediante un mando sencillo[3]. Esta se compone de una serie de botones de colores, directamente relacionados con los del mando (figura 5.2), que permiten iniciar una comunicación mediante videoconferencia con el personal terapéutico como finalizar la reunión. Durante el proceso de la llamada, el vídeo del paciente es capturado y enviado al servidor para su procesado posterior.

La segunda parte consiste en la interfaz del terapeuta encargado de dirigir las sesiones de rehabilitación del paciente ofreciendo una interfaz sencilla que permite conectarse a las videoconferencias de los diferentes

[3]Concretamente un mando de la consola SNES adaptado para ser usado mediante USB

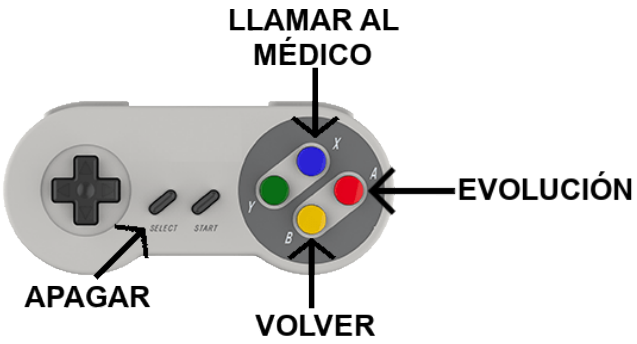

Figura 5.2: Funcionalidades del mando de SNES para el control de la aplicación web por parte del paciente.

Figura 5.3: Menú del terapeuta

pacientes y dirigir las sesiones de terapia además de gestionar la evolución del paciente (figura 5.3).

Las características del equipo donde está desplegada la aplicación del paciente es:

- MSI - Cubi N 8GL-001BEU N4000 1.10GHz.
- WB Green M.2 120 GB SATA 3.
- X GB DDR4 2400 MHz.
- Lubuntu x86_64 18.04.
- WebCam Logitech HD Pro C920

TODO: Quizás sería interesante explicar las valoraciones otorgadas de la aplicación si se tienen antes de la entrega del trabajo.

Trabajos relacionados

Conclusiones y Líneas de trabajo futuras

Apéndices

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

La planificación temporal se ha realizado adaptando la metodología *Scrum*. Para poder adaptarlo a un trabajo de una sola persona para un proyecto educativo se han han considerado las siguientes indicaciones:

- El desarrollo se ha basado en iteraciones o *sprints* de dos semana de duración aproximadamente.
- Cada uno de los *sprints* contiene las tareas que se realizaron en el mismo.
- Cada tarea tiene un coste estimado dependiendo de lo que el programador estime conveniente siguiendo los parámetros de tiempo a emplear, dificultad técnica entre otros.
- Una vez concluida una tarea se especifica el coste real para poder estimar de una manera más correcta tareas de *sprints* siguientes.
- Al finalizar cada *sprint* se realiza una reunión con los tutores del proyecto.

Sprint 0

El *sprint* 0 consistió en el desarrollo de la aplicación web para la recogida de datos para el proyecto. Es el único *sprint* realizado en colaboración con el alumno José Miguel Ramírez Sanz.

Tarea	**Estimado**	**Final**
Diseño de la interfaz web	3	3
Creación de la plantilla maestra de toda la web	3	2
Creación de la plantilla base para el menú del paciente	5	5
Implementación del menú del terapeuta	2	2
Creación de la conexión para videollamada - Terapeuta	1	1
Creación de la conexión para videollamada - Paciente	1	1
Implementación del sistema de inicio de sesión	2	3
Creación de *plugin* para la captura y emisión del vídeo del paciente	8	13
Creación de la interfaz de gestión del paciente	2	2

Tabla A.1: Tareas del *sprint* 0

Sprint 1

El *sprint* 1 consistió en la exploración de herramientas para la creación y procesado de flujos de vídeo.

Tarea	**Estimado**	**Final**
Búsqueda de herramientas para la creación de flujos de datos	2	2
Pruebas sobre *Apache Flume*	5	3
Pruebas sobre *Apache Kafka*	5	5
Búsqueda de herramientas para el despliegue por contenedores	2	2
Prueba de despliegue de *Apache Kafka* para *Docker*	2	2
Búsqueda de herramientas para el procesado de flujos de vídeo	2	2
Pruebas con *Spark Streaming* con *OpenCV*	5	13

Tabla A.2: Tareas del *sprint* 1

Sprint 2

El *sprint* 2 consistió en la implementación de las conexiones entre todos los elementos del flujo a nivel local.

Tarea	**Estimado**	**Final**
Despliegue local de *Apache Spark*	1	1
Despliegue local de *Apache Kafka*	3	5
Simulación de servidor UDP para vídeo	5	5
Ingestor a Kafka del vídeo UDP	3	5
Conectar *Spark Streaming* con *Kafka*	3	13
Implementar un anonimizador de rostros	3	2
Parametrizar todos los *scripts* creados	1	1

Tabla A.3: Tareas del *sprint* 2

La razón de la gran diferencia en la quinta tarea del *sprint* entre predicho e invertido fue debido a que la documentación de *Kafka* no estaba bien detallada para conectar con *Spark Streaming*.

Sprint 3

El *sprint* 3 consistió en la implementación de una infraestructura de contenedores *Docker* que conectase todos los servicios para el flujo.

Tarea	**Estimado**	**Final**
Despliegue de *Apache Spark* para el nodo *máster*	2	2
Despliegue de *Apache Spark* para varios nodos *slave*	2	1
Despliegue de la aplicación *openCV*	5	3
Recogida de *stream* de vídeo e ingestión en *Kafka*	8	8

Tabla A.4: Tareas del *sprint* 3

Sprint 4

El *sprint* 4 consistió en la automatización de los procesos del *sprint* 2[1].

[1]Pendiente de realizar

Tarea	Estimado	Final

Tabla A.5: Tareas del *sprint* 4

Sprint X

El *sprint* X consistió en Y[2].

Tarea	Estimado	Final

Tabla A.6: Tareas del *sprint* X

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

[2]Plantilla

Apéndice B

Especificación de Requisitos

B.1. Introducción

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] Apache Flume. https://flume.apache.org/.

[2] Apache Hadoop. https://hadoop.apache.org/.

[3] Apache Kafka. https://kafka.apache.org/.

[4] Spark Streaming - Spark 2.4.5 Documentation. https://spark.apache.org/docs/latest/streaming-programming-guide.html.

[5] Mario Juez-Gil. mjuez/spark-cluster-docker. https://github.com/mjuez/spark-cluster-docker.

[6] MuyLinux. Red Hat lidera el segmento Linux en el mercado de servidores. https://www.muylinux.com/2018/10/19/red-hat-lidera-mercado-linux-servidores/.

[7] Wurstmeister. wurstmeister/kafka-docker. https://github.com/wurstmeister/kafka-docker.

[8] Wensong Zhang et al. Linux virtual server for scalable network services.