Tarea 1 - Servicios en contenedores

Jung-Hwan Bak, 2016193299*
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica
Email:*bakkim05@gmail.com

I. Introducción

En esta tarea se crea una imagen de Docker que recibe imágenes utilizando sockets en un modelo cliente - servidor utilizando el lenguaje de programación C. Cuando el servidor recibe los datos de la imagen, será procesada por medio de un algoritmo para detectar su componente de color principal (R,G,B). Una vez se detecte cual es el componente de color principal, la imagen será guarda en carpetas respectivas con su color. Para el procesamiento de imágenes seguras, el servidor tendrá una lista de IPs permitidas que puedan enviar imágenes a ser procesadas. Si un IP no permitido envía una imagen, el archivo no será categorizado por el algoritmo y se pondrá en una carpeta aparte.

En la segunda sección, se discute los detalles relacionados a la implementación y las herramientas utilizadas durante el proyecto. La tercera sección, se ubica la descripción de los atributos reforzados durante el desarrollo del proyecto. En la cuarta sección se ubica las descripciones necesarias para la comprensión del diseño y funcionamiento del proyecto. En la quinta sección se describe las instrucciones para la utilización correcta del proyecto. La bitácora de actividades se muestra en la sexta sección. Las conclusiones se ubican en la séptima sección, las sugerencias en la octava y se concluye el documento con la referencias.

II. Ambiente de Desarrollo

El programa fue creado utilizando el sistema operativo Ubuntu 20.04.1 LTS, principalmente Visual Studio Code versión 1.49.2 como editor de texto, gcc versión 9.3.0 para la compilación del cliente-servidor, la biblioteca stb_image (stb_image.h versión 2.26 y stb_image_write.h versión 1.15) para el procesamiento de imágenes en el algoritmo de color y Docker versión 19.03.12 para la creación de contenedores utilizados para el servidor. La imagen dentro del contenedor de Docker tiene centOS 8 como sistema operativo.

A. Algoritmo de color

El algoritmo es desarrollado utilizando la biblioteca stb_image en C. El algoritmo recorre píxel por píxel y determina cual de los componentes RGB es el más alto, en caso de ser un píxel blanco (255,255,255) o negro (0,0,0) se clasifica como rojo. Una vez se determina cual de los componentes es el más alto, se agrega al contador respectivo del color. Al finalizar el recorrido se tendrá cual de los componentes RGB es el que domina la imagen.

B. Archivo de configuración y Comunicación

El archivo de configuración contiene una lista de IPs que pueden utilizar el algoritmos de colores del servidor. Este archivo tiene un IP por linea y tiene el nombre de ”configuracion” con extensión ”.config”. Este archivo es parseado dentro del servidor el cual modifica un booleano (valor base falso) a verdadero en caso de que se detecte una coincidencia del IP del cliente y los IPs dentro del configuracion.config.

C. Las carpetas

Las carpetas en las cuales las imágenes recibidas van a ser guardadas son creadas por medio del Dockerfile y son mantenidas por medio de un volumen en la cual esta montado el contenedor. Existen cuatro carpetas diferentes: red, green, blue y Not_Trusted. En la carpeta Not_Trusted van las imágenes en las cuales el whitelist del IP con se pudo realizar y en las demás van respectivo al color determinado por el algoritmo de color.

D. Cliente

El cliente recibe como parámetro el IP del servidor a la cual se quiere conectar utilizando el puerto 8080. Se crea un socket con el protocolo de comunicación IPv4 y se utiliza TCP como tipo de comunicación. Una vez se establece la conexión con el servidor, el cliente introduce por medio de la terminal el directorio de la imagen que desea enviar al servidor. Si se desea finalizar la conexión con el servidor, el usuario puede introducir la palabra ”fin” como directorio de imagen.

III. Atributos

A. Aprendizaje Continuo

Para la elaboración del cliente y el servidor se utilizo conocimientos adquiridos en los cursos de Algoritmos y Estructuras de Datos 1 y Algoritmos y Estructuras de Datos 2. El desarrollo del parser para el archivo de configuracion.config fue un conocimiento adquirido por medio del desarrollo de un compilador en la clase de Lenguajes, compiladores e interpretes y la implementación de del algoritmo color fue utilizando conocimientos adquiridos con el tiempo. En esta tarea se aprendió sobre la implementación de Docker en un ambiente de desarrollo para la containerization del servidor y sobre sus ventajas sobre las formas tradicionales de virtualización.

B. Herramientas de Ingeniería

En esta tarea se aprende sobre el containerization utilizando la herramienta de Docker. El objetivo de Docker es facilitar la creación, el despliegue y la entrega de una aplicación utilizando los llamados Contenedores. Los contenedores permiten al desarrollador agrupar una aplicación con todos los componentes necesarios (librerías y otros recursos) y entregarla como un paquete independiente y único [1]. La herramienta de Docker fue utilizado para crear un contenedor con una instancia del servidor para darle uso a la containerization que está siendo utilizado en la industria debido a sus grandes ventajas [1].

IV. Detalles del diseño del programa desarrollado

En la Fig.1 se puede observar un diagrama de alto nivel que describe el comportamiento que tiene la interacción entre el cliente-servidor y las acciones que toma el servidor una vez recibe una imagen. La imagen es transmitida por medio de sockets hacia el servidor. El recibe la conexión con el cliente y determina si el IP del cliente esta dentro de la lista de los admitidos. Si el IP es admitido, el servidor procederá a clasificar las imágenes por su color dominante y guardarlo en su carpeta respectiva, de lo contrario, será puesto en un archivo aparte sin que la imagen sea procesada.

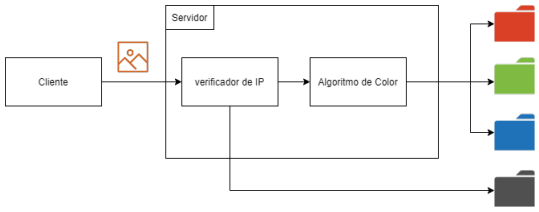

Fig. 1. Diagrama de alto nivel

V. Instrucciones de uso

1) Ir al archivo de operativos_tarea_1.
2) Abrir la terminal dentro del directorio.
3) Correr el comando: make.
4) Ingresar a la carpeta clientside.
5) Abrir la terminal dentro del directorio.
6) Correr el comando: ./client [IP del servidor].
7) Introducir el directorio de la imagen que desea enviar.
8) Introducir la palabra ”fin” como directorio de la imagen a enviar para terminar la conexión con el servidor.

VI. Tabla de actividades

Actividad	Horas
Investigación	6
Algoritmo Color	2
Whitelist	1
Client & Server	4
Docker	4
Documentación	2

TABLE I
TABLA DE ACTIVIDADES: BITÁCORA CON EL TOTAL DE HORAS TRABAJADAS.

VII. Conclusiones

Los contenedores funcionan un poco como los VM, pero de una manera mucho más específica y granular. Aíslan una sola aplicación y sus dependencias, todas las bibliotecas de software externo que la aplicación requiere para funcionar, tanto del sistema operativo como de otros contenedores.

VIII. Sugerencias y Recomendaciones

- Utilizar comunicaciones del tipo TCP a la hora de transferir imágenes por medio de sockets.
- Utilizar la biblioteca STB_IMAGE para la manipulación de imágenes, debido a que es una biblioteca muy liviana. Carece de una documentación extensa.

References

[1] “Why Docker is so important? — Clickode,” Clickode.com, 2020. [Online]. Available: http://www.clickode.com/en/2016/01/26/why-docker-important [Accessed: 25-Sep-2020].