Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 6, No. 2, pp. 21-29, Diciembre 2019
Fecha recepción: 14/12/2019, Fecha aceptación: 10/05/2020

Extendiendo ZoneMinder para su Uso Masivo con una Refactorización de las Interfaces de Administración y la Integración de un Módulo de Reconocimiento Facial

Ilich Rondon[1], Kenny Jimenez[1], Dedaniel Urribarri[1], Eric Gamess[2]
ilichrondon@gmail.com, kenny.jimenez05@gmail.com, dedanielu@gmail.com, egamess@jsu.edu

[1] Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela
[2] MCIS Department, Jacksonville State University, Jacksonville, AL, USA

Resumen: A través de los años, los requerimientos de monitoreo y control de sitios de importancia en empresas, instituciones, e inclusive hogares, se han incrementado. La solución inicial fue la implementación de CCTV (Circuitos Cerrados de TeleVisión) basados en cámaras analógicas y restringida a usuarios pudientes. Además de los altos costos, esta solución carecía de sensores de movimientos. Frente a estas limitaciones, surgió una alternativa conocida como video vigilancia IP que usa la red de datos preexistente. Debido al auge de la video vigilancia IP, se hace necesario un software libre, con capacidad de administración centralizada para cámaras IP, con pocos requerimientos de cómputo, y una interfaz sencilla y amigable que permita visualizar las proyecciones y configurar de forma remota los parámetros de monitoreo. En este trabajo, se desarrolló un NVMS (Network Video Management System) basado en software abierto, con reconocimiento facial, que pueda ser usado por usuarios con pocos conocimientos en el área, como una solución de monitoreo de espacios pequeños o medianos. Para validar la poca carga del sistema propuesto, se hicieron pruebas de validaciones en varios escenarios.

Palabras Clave: Video Vigilancia; NVMSs; ZoneMinder; Cámaras IP; Reconocimiento Facial; Código Abierto; Administración Centralizada.

Abstract: Over the years, the requirements to monitor and control important sites in companies, institutions, and even homes, have increased. The initial solution was the implementation of CCTV (Closed-Circuit TeleVision) based on analog cameras and restricted to wealthy users. In addition to the high costs, this solution lacked motion sensors. Given these limitations, an alternative emerged which is known as "IP video-surveillance" that uses the pre-existing data network. Due to the increase in IP video-surveillance, a free software is needed, with centralized administration capacity for IP cameras, with low computational requirements, and a simple and friendly interface that allows viewing projections and remotely configuring monitoring parameters. In this work, a NVMS (Network Video Management System) based on free software was developed, with facial recognition, that can be used by users with little knowledge in the area, as a monitoring solution for small or medium spaces. To validate the low load of the proposed system, validation tests were performed in various scenarios.

Keywords: Video-Surveillance; NVMSs; ZoneMinder; IP Cameras; Facial Recognition; Open Source Code; Centralized Management.

I. Introducción

La video vigilancia [1-5] se originó como una solución a la necesidad de mantener constantemente vigilado un lugar y se popularizó con los primeros Circuitos Cerrados de Televisión (CCTV) [2] con cámaras analógicas. En sus inicios, la video vigilancia era destinada únicamente a ambientes empresariales y gubernamentales. Sin embargo, con el creciente desarrollo tecnológico de las últimas décadas, la significativa bajada de los precios de las cámaras IP y la facilidad de crear aplicaciones de software, el uso de la video vigilancia se ha extrapolado a cualquier lugar que necesite una solución eficiente de seguridad, inclusive lugares como el hogar.

Con el pasar de los años, la video vigilancia IP ha ido involucrando y agregando un conjunto de herramientas y tecnologías que han mejorado el control de seguridad que se puede tener de un lugar a través de las cámaras IP. Tecnologías tales como la detección de movimiento por medio de sensores, o incluso la agregación de algoritmos para el reconocimiento facial han proporcionado a los sistemas de administración de

cámaras IP grandes avances en materia de monitoreo y seguridad.

Debido a la gran demanda que existe actualmente sobre los softwares de administración de cámaras IP en el ámbito de la seguridad, y la capacidad que estos poseen para poder aplicar avanzadas tecnologías como lo es el reconocimiento facial con el análisis de las imágenes y videos recolectados, es necesario que estos posean interfaces de administración que sean fáciles e intuitivas de utilizar. El objetivo final es que todos los entes interesados en aplicar este tipo de sistemas de seguridad se vean respaldados al momento de implementarlos, debido a que no van a necesitar de personal especializado, y se van a beneficiar de las importantes funcionalidades de monitoreo y seguridad que estos ofrecen en el área de la video vigilancia.

En este trabajo, se implementó un sistema de video vigilancia basado en softwares de código abierto como lo son ZoneMinder [6][7] y el framework de Python llamado Django [8-10]. La idea principal es el uso de ZoneMinder como un sistema para configurar y explotar cámaras IP, con una refactorización de sus interfaces de administración para que la gestión del sistema y de los sensores sea al alcance de todos, además de la integración de un módulo de reconocimiento facial. Para validar los pocos recursos requeridos por el sistema desarrollo y su adecuado rendimiento para entidades pequeñas o medianas, se hicieron pruebas de desempeño para las cuales se reportan (1) el uso de la memoria RAM, (2) la carga promedia, (3) el uso del CPU y (4) el tiempo de respuesta, en diferentes escenarios de pruebas. Nuestras pruebas indican que el sistema propuesto en este trabajo es adecuado para dichas entidades.

El resto del presente documento está organizado como especificado a continuación. Los trabajos relacionados son revisados en la Sección II. La Sección III presenta la arquitectura y el diseño de la solución. Se discuten las pruebas de estrés y el análisis de los resultados obtenidos en la Sección IV. Finalmente, se presenta un balance del trabajo en la Sección V a través de las conclusiones y se dan direcciones para trabajos futuros en la Sección VI.

II. Trabajos Relacionados

Existen muchos proyectos comerciales para la televigilancia basada en IP, como por ejemplo Xeoma [11] de Felenasoft, Kerberos [12] de Verstraeten, XProtect [13] de Milestone Systems, EyeLine [14] de NCH Software, ContaCam [15] de Contaware.com, iVideon [16] de Mobile Video Solutions Inc., Blue Iris [17] de Blue Iris Software, Evo S [18] de Luxriot, webcamXP [19] de Moonware Studios, SmartViewer [20] de Hanwha Techwin Europe, y Camcloud VMS [21] de Camcloud.

Xeoma [11] es un software comercial de video vigilancia que permite la administración de decenas de cámaras desde un único servidor. El programa es multiplataforma y funciona en Windows, Linux, MacOS, y dispositivos Android e iOS, con acceso remoto completo y vista desde cualquier dispositivo móvil.

Xeoma ofrece auto-detección y soporte para casi cualquier cámara IP, como las basadas en ONVIF [22] (Open Network Video Interface Forum, un foro abierto industrial que provee y promueve interfaces estándares para una interoperabilidad efectiva de productos de seguridad basados en IP), y las webcams USB (hasta 99,9% de las cámaras del mercado son soportadas). Con su ayuda, en tan sólo unos segundos, un computador y una cámara se convierten en un sistema de vigilancia. También posee un detector de movimiento con funciones avanzadas de falsas alarmas, evitando así la activación de la lógica cuando no es necesario. Igualmente, incluye notificaciones programadas (SMS, correo electrónico, sonido de alarma, entre otras). A diferencia de otros productos en el mercado, Xeoma también ofrece el monitoreo del audio obtenido a través de las cámaras.

Kerberos [12] es un software de video vigilancia que tiene una versión de código abierto (Kerberos.io) y una versión comercial (Kerberos Enterprise). La versión libre de costo es más limitada, por ejemplo, funciona bien cuando el número de cámaras es bajo. La versión comercial, Kerberos Enterprise, escala bien con docenas o centenas de cámaras y es de alta disponibilidad. Kerberos Enterprise está basada en Kubernetes [23][24]. En el caso de Kerberos.io, existen varias posibilidades como un Docker o la instalación en SBCs (Single Board Computer) como Raspberry Pi para un mejor soporte de las aplicaciones IoT (Internet de las Cosas).

Kerberos.io está provisto de una interfaz limpia y ordenada, y tiene soporte para una gran cantidad de cámaras. Puede usar cámaras USB, IP o Raspberry Pi (v1.3 y v2.1). Cuando utiliza cámaras Raspberry Pi, se beneficia de su codificación de hardware que permite grabar videos a 30 FPS en un Raspbery Pi Zero. Cuando se desea utilizar una cámara IP, se recomienda usar la conexión RTSP [25-27] (Real Time Streaming Protocol), si está disponible. Se debe tener en cuenta que es posible que las cámaras IP económicas no funcionen correctamente. En forma general, las cámaras USB que necesitan controladores especiales no funcionan.

Ciertos proyectos son totalmente basados en soluciones orientadas a la nube, como Camcloud VMS [21]. Con Camcloud VMS, lo único que el cliente necesita comprar al nivel de hardware son las cámaras IP. El software de administración corre en la nube y autodetecta las cámaras para una configuración sencilla. La nube almacena todas las grabaciones críticas hasta 90 días, y los usuarios son capaces de acceder y manejar sus registros desde el sitio web de Camcloud o desde una aplicación móvil. La empresa también ofrece una opción adicional para almacenamiento local para mayor redundancia, de tal forma que, si la conexión entre las cámaras y el Internet se pierde, todo se graba localmente y puede ser revisado a posteriori.

El precio de las soluciones comerciales depende del número de cámaras a conectar, de las opciones escogidas (e.g., detección de movimiento, reconocimiento facial) y del almacenamiento requerido en la nube. Muchos de estos productos tienen versiones gratis, pero muy limitadas. Entre las limitaciones más comunes de las versiones de evaluación, se pueden citar: (1) el número de cámaras que se puede conectar, (2) el almacenamiento en la red, (3) el reconocimiento facial, y (4) la disponibilidad del código fuente. A no ser de código abierto, los usuarios no tienen la posibilidad de agregar o modificar funcionalidades, a gusto.

Si existen centenas de soluciones comerciales, a la fecha, el mundo del código abierto está mucho más limitado, y los grandes nombres incluyen iSpy [28] de iSpyConnect.com, Bluecherry [29] de Bluecherry, Shinobi [30] de Moe Alam,

MotionEyes [31] de Calin Crisan, y ZoneMinder [6][7] soportado por una gran comunidad.

iSpy [28] es un sistema de video vigilancia de código abierto desarrollado en C# que funciona solo en PC con Windows (versiones de 32 y 64 bits). Se sabe que funciona bien en Windows 7/8/10, pero pudiese funcionar en otras variantes de Windows que soporten el .NET Framework v4.5, o superior. iSpy es compatible con la gran mayoría de cámaras web y cámaras IP. El uso más común de iSpy es la seguridad en empresas pequeñas, pero el monitoreo del hogar, la vigilancia del vecindario y el control de los niños son también características valiosas. Con iSpy, los usuarios pueden definir áreas específicas del video que se deben monitorear para ver si hay movimiento y establecer un valor umbral para la cantidad de movimiento que desencadenaría la grabación automática. No hay limitaciones en cuanto al número de sensores (micrófonos o cámaras) que se pueden conectar al software. iSpy se puede descargar y usar de forma gratuita localmente, pero acceder a las cámaras de forma remota, las alertas por SMS, Twitter o emails requieren una suscripción paga.

Bluecherry [29] es una aplicación de código abierto de video vigilancia para Linux (Debian, Ubuntu y CentOS) que soporta cámaras IP. El software hace un bajo uso de memoria, es compatible con ONVIF [22] y toma ventaja del GPU para la detección de movimiento. Esto significa que un servidor de baja gama con un GPU soporta múltiples flujos provenientes de cámaras, con un bajo uso del CPU. Para la configuración y el monitoreo en tiempo real, Bluecherry se puede acceder tanto a través de una interfaz web como a través de un cliente de código abierto soportado en múltiples sistemas operativos (Linux, Windows y MacOS X). El servidor está principalmente escrito en C/C++, con algunos scripts desarrollados en PHP. El cliente está programado en C++ y hace uso de la librería Qt [32].

Shinobi [30] es un servidor de video vigilancia de código abierto escrito en Node.js. Shinobi se basa en FFmpeg [33] (una solución multiplataforma para grabar, convertir y transmitir audios y videos), MariaDB y Node.js, y utiliza masivamente JavaScript, un poco de Python, y algo de Bourne Shell. El servidor es multiplataforma (BSD, Linux, MacOS, Windows) y compatible con la arquitectura ARM, además que también se puede utilizar con una imagen en Docker. Shinobi permite recuperar secuencias de audio y video de cámaras a través de HTTP, RTP/RTSP y ONVIF [22], se admite HTTPS, pero solo con certificados X.509 válidos. En términos de funcionalidad, Shinobi está en algún lugar entre ZoneMinder (utilizable en un entorno profesional y antiguo) y Kerberos.io.

MotionEyeOS [31] es una distribución de Linux que convierte un SBC (Single Board Computer) en un sistema de video vigilancia. El sistema operativo se basa en Buildroot [34-36] y utiliza Motion como backend y MotionEye como frontend.

- Buildroot es un conjunto de Makefiles y parches que simplifica y automatiza el proceso de construcción de un entorno Linux completo y de arranque para un sistema embebido. Utiliza la compilación cruzada para permitir la construcción de múltiples plataformas objetivo (target platforms) en un solo sistema de desarrollo basado en Linux. Buildroot puede construir automáticamente la cadena de herramientas de compilación cruzada requerida, crear un sistema de archivos raíz, compilar una imagen del kernel de Linux y generar un cargador de arranque para el sistema embebido, o puede realizar cualquier combinación independiente de estos pasos. Por ejemplo, una cadena de herramientas de compilación cruzada ya instalada se puede usar de forma independiente, mientras que Buildroot solo crea el sistema de archivos raíz.
- Motion es un programa altamente configurable que monitorea las señales de video de muchos tipos de cámaras. Puede ser utilizado para crear videos o guardar fotos de las actividades capturadas, grabar desde múltiples cámaras IP, ver transmisión en vivo de cámaras, ejecutar scripts cuando ocurren actividades, registrar actividad en múltiples tipos de bases de datos, y dar soporte completo de TLS (HTTPS) con autenticación para control web y transmisiones. Motion es compatible con muchos tipos de dispositivos y protocolos tales como cámaras de red a través de RTSP, RTMP y HTTP, cámaras web V4L2, tarjetas de captura de video, y archivos de películas existentes, entre otros.
- MotionEye es un frontend web para el demonio Motion, escrito y desarrollado bajo el lenguaje Python.

ZoneMinder [6][7] es un conjunto integrado de aplicaciones de código abierto desarrolladas en C++, Perl y PHP que provee una solución completa de video vigilancia, permitiendo la captura, el análisis, la grabación y el monitoreo a través de cámaras de seguridad, conectadas a un sistema Linux o FreeBSD. Fue diseñado para ser ejecutado en distribuciones que soporten la interfaz V4L (Video For Linux) y puede trabajar con una gran variedad de cámaras que incluyen tanto las cámaras USB y FireWire, como las cámaras IP.

ZoneMinder es una opción popular para la video vigilancia de casas como de negocios. Trabaja simultáneamente con varias cámaras. La grabación puede empezar cuando la aplicación detecta cambios entre frames, y el usuario puede definir zonas de las vistas que deben ser ignoradas. ZoneMinder es seguramente el proyecto de código abierto más avanzado y utilizado en la comunidad. Sin embargo, es bien conocido que la configuración de las cámaras no es sencilla, y la documentación de ZoneMinder a este nivel deja mucho que desear.

Eusebio y Mello [37] presentaron una solución para teléfonos inteligentes Android, llamada DroidMinder, que consiste en una aplicación cuyo objetivo es actuar como agente cliente para un sistema de vigilancia ZoneMinder. Es de recordar que si ZoneMinder ofrece un agente propio con una interfaz web simplificada, este no fue pensado para las nuevas tecnologías de teléfonos móviles, motivando este desarrollo.

En ZoneMinder, la información de las cámaras, los usuarios y las alarmas se almacenan en una base de datos relacional MySQL. El cliente predeterminado de ZoneMinder consiste en una página en PHP que accede directamente a esta base de datos, disfrutando de la facilidad de estar ejecutándose en la misma máquina donde se encuentra la base de datos. El objetivo de DroidMinder es permitir a los usuarios monitorear sus cámaras de vigilancia a través de Internet, estando esta aplicación desacoplada físicamente de la máquina donde está en ejecución ZoneMinder. El grupo de investigadores exploraron dos opciones para el diseño de DroidMinder:

- Desarrollar un agente cliente que realice consultas directas a la base de datos MySQL.
- Desarrollar un agente cliente que accede a la interfaz web de ZoneMinder y obtenga los datos multimedia a partir de los flujos de video de las cámaras de vigilancia.

Los autores de DroidMinder optaron por el segundo enfoque por el hecho de ser más transparente para el usuario, pues bastaría proporcionar al DroidMinder la dirección IP y puerto donde se está ejecutando ZoneMinder, para así obtener acceso a las cámaras de vigilancia conectadas a la red. De esta forma, no hay necesidad de realizar modificaciones en la base de datos MySQL, o incluso crear nuevas reglas en el firewall.

En el laboratorio de Investigación Imagelab de la Universidad de Modena y Reggio Emilia, Italia [38], se llevó a cabo un proyecto llamado ViSERaS (Video Surveillance in Emilia Romagna as a Service) entre los años 2010 y 2011. El objetivo del proyecto fue desarrollar un sistema integrado de administración y análisis de videos como un servicio, donde las entidades públicas (en particular en los pequeños municipios de la Región de Emilia-Romagna) podrían disponer de la gestión, almacenamiento y análisis de un gran número de cámaras, a través de un SaaS, aprovechando el ancho de banda proporcionado por Lepida SpA.

Uno de los requisitos más importantes del proyecto era emplear, especialmente para el sistema de Video Management, soluciones de código abierto. Entre las muchas soluciones posibles hoy en día, el laboratorio de investigación eligió ZoneMinder porque exhibe varias características interesantes, tales como:

- Admite varios tipos de cámaras, incluida PTZ.
- Está basado en herramientas de programación estándares como C ++, Perl y PHP.
- Usa un manejador de bases de datos de código abierto (MySQL) y de alto rendimiento.
- Admite videos en vivo en MPEG y JPEG de varias partes e imágenes fijas.

Es de aclarar que, si el reconocimiento facial es una opción bastante común en los softwares comerciales, los desarrollos de dominio público se limitan a la detección de movimientos. Además, en muchos softwares de código abierto, la configuración se debe hacer a través de archivos de configuración y reiniciando los demonios asociados, lo que hace su administración más compleja que los software comerciales. Estas razones motivaron este desarrollo e investigación.

III. Arquitectura y Diseño de la Solución

En la Figura 1, se puede observar las fases establecidas para llevar acabo la solución propuesta. También se muestra, de forma genérica, cómo es el flujo básico del sistema, específicamente él del módulo de reconocimiento facial. A continuación, se especifican cada una de las fases involucradas en la Figura 1.

A. Refactorización de las Interfaces Web para la Administración de ZoneMinder

Parte de la propuesta consiste en la refactorización de las interfaces para el sistema de administración de cámaras IP ZoneMinder. ZoneMinder fue la opción más favorable según la investigación llevada a cabo (ver Sección II) para la refactorización de las interfaces web y la integración de un módulo de reconocimiento facial. Es de aclarar que además de ser de código abierto, ZoneMinder dispone de una buena documentación al nivel de desarrollo, que permite entender y trabajar con el código fuente, facilitando así su modificación para adaptarlo a las necesidades específicas.

La restructuración de las interfaces fue enfocada al mejoramiento de la experiencia de administración de las cámaras a través del sistema propuesto. Además de una mejor experiencia para la administración, el sistema desarrollado permite al usuario tener un excelente control de todas las funcionalidades, inclusive el ingreso desde cualquier tipo de dispositivo (teléfono inteligente, tablet, laptop, o computador).

B. API RESTful para el Reconocimiento Facial

El núcleo principal de la segunda fase fue en el diseño y la implementación de un API RESTful relacionado con el reconocimiento facial, basado en un framework de Python llamado Django [8-10]. El API desarrollado ofrece funciones que incluyen (1) el registro o enrolamiento de usuarios, (2) la carga de imágenes del rostro de los usuarios con diferentes

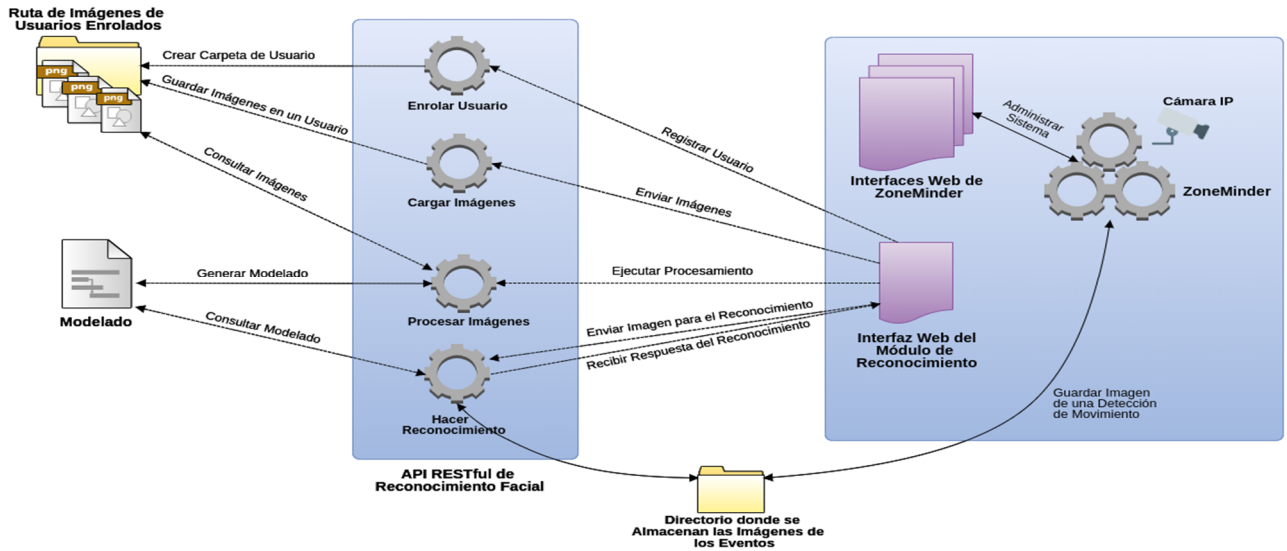

Figura 1: Arquitectura de la Solución

ángulos, (3) la ejecución del procesamiento de imágenes para generar la base de conocimiento para el reconocimiento de los usuarios, y (4) la realización de peticiones de reconocimiento de un usuario, dado una imagen.

C. *Módulo de Reconocimiento Facial*

Este módulo fue integrado como una interfaz web en el sistema de administración de ZoneMinder. La fase de reconocimiento facial, o segunda fase, está constituida por el API y el módulo de reconocimiento facial. A través de este módulo, el usuario tendrá control de todas las funcionalidades presentes en el API.

IV. Pruebas de Rendimiento y Analisis de Resultados

Las pruebas se realizaron estresando directamente el API RESTful sin necesidad de utilizar el módulo de reconocimiento facial que fue integrado en ZoneMinder. Para llevar a cabo las pruebas de rendimiento y de estrés, se utilizó JMeter [39-42], un proyecto de Apache que puede ser utilizado como una herramienta de prueba de cargas para analizar y medir el desempeño de diversos servicios, con énfasis en aplicaciones web. La herramienta de JMeter con la cual se ejecutaron las pruebas corrió en una máquina real (i.e., no en un ambiente virtualizado), fuera del ambiente donde se desplegó la solución. Esta máquina contaba con 16 GB de RAM, un disco duro de 500 GB, un procesador Intel i5-3210M @ 2.50 GHz con 4 núcleos y Ubuntu 18.04 como sistema operativo.

Para implementar el sistema a probar, se utilizó un cluster de tres servidores donde se instaló el hipervisor Proxmox VE [43-46], con las siguientes características agregadas entre los tres servidores:

- 80 núcleos de procesamiento
- 500 GB de memoria RAM
- 5 TB de disco duro

Proxmox VE [46] es una plataforma de código abierto para virtualización que integra el hipervisor KVM [47][48], los contenedores LXC [49], el almacenamiento definido por software (Software-Defined Storage), y funcionalidades de redes en un único sistema, con alta disponibilidad y sencillo de administrar a través de interfaces web. Sobre este marco de virtualización, se creó una máquina virtual con CentOS 7.8 como sistema operativo, donde fue desplegada la solución. La máquina virtual fue configurada con 8 núcleos de procesamiento, 16 GB de RAM y un disco duro de 20 GB.

Se realizaron dos sets de pruebas de estrés para evaluar diferentes aspectos del sistema propuesto cuando se encuentra sometido a una fuerte carga. Para las pruebas, se disponía de un conjunto de imágenes almacenadas en el sistema, obtenidas desde una cámara IP conectada a ZoneMinder, que correspondían a usuarios conocidos. En el primer set de pruebas realizadas, se mantuvo fijo el número de imágenes y se varió el número de peticiones al servicio, como es especificado en la Tabla I. Es importante mencionar que el Rampup de JMeter para esta y todas las demás pruebas fue de 1 segundo. Esto significa que todas las peticiones se dispararan de forma concurrente durante el primer segundo del experimento. Un número de 20 peticiones en un segundo se acerca a la máxima carga que podría tener un hogar. Por otro lado, 160 peticiones en un segundo corresponderían a la máxima carga que se debería observar en una entidad mediana.

Tabla I: Datos Asociados al Set de Pruebas #1

Peticiones	20	40	60	80	100	120	140	160
Imágenes	50	50	50	50	50	50	50	50

Durante la ejecución de las pruebas, se recolectó un conjunto de datos sobre el uso de los recursos del ambiente, empleando la versión de código abierto de Grafana [50], una plataforma para el monitoreo y la observación de experimentos. Con estos datos, se realizaron una serie de gráficas que muestran el comportamiento y el uso de los recursos en la máquina virtual durante la ejecución de las pruebas (uso de memoria RAM, carga promedio, uso del CPU y tiempo de respuesta).

En la Figura 2, se puede observar el uso de la memoria RAM en la máquina virtual para las pruebas del primer set. Por cada experimento, se obtuvieron los valores mínimo, promedio y máximo del uso de la memoria RAM. Se puede observar que, con un número de 50 imágenes y un conjunto variable de peticiones al servicio, el punto más alto de consumo de memoria RAM fue de 11.7 GB. Eso es, un computador mediano con 16 GB debería poder soportar sin problema los requerimiento de memoria RAM.

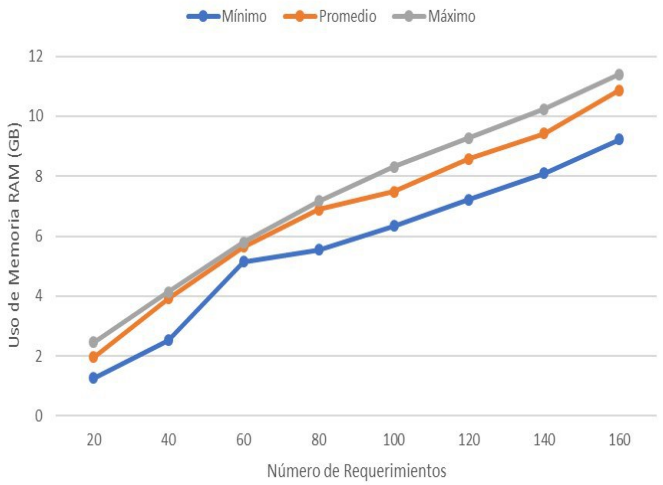

Figura 2: Uso de Memoria RAM para el Set de Pruebas #1

La Figura 3 muestra la "carga promedio" del sistema durante la ejecución de las peticiones, que corresponde al número promedio de procesos a ejecutarse en un tiempo dado, mientras que se ejecutaban peticiones. Se puede observar que la mayor carga registrada durante el primer set de pruebas fue de 56. Como la máquina virtual usada tenía 8 núcleos de procesamiento, esto quiere decir que un momento dado, hubo 48 procesos pendientes por ejecutarse en el sistema. En otras palabras, en el peor periodo de peticiones, se generó un encolamiento de hasta 48 procesos. Sin embargo, la curva promedio de la "carga promedio" revela que cuando se hicieron 160 peticiones, el número promedio de procesos fue de 34.

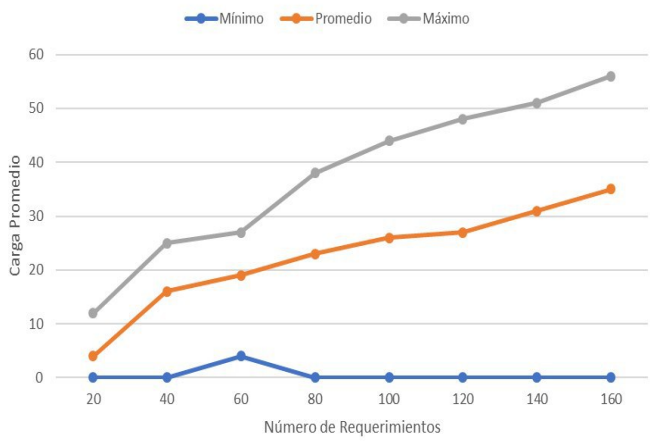

Figura 3: Carga Promedio para el Set de Pruebas #1

En cuanto al uso del CPU en la Figura 4, podemos observar que a medida que aumenta el número de peticiones, también aumenta el uso del CPU. Con la curva promedio, se puede observar que después de 50 peticiones, el CPU tiene un uso de por lo menos 80%, que se podría considerarse como la carga máxima a la cual se debería someter esta máquina virtual.

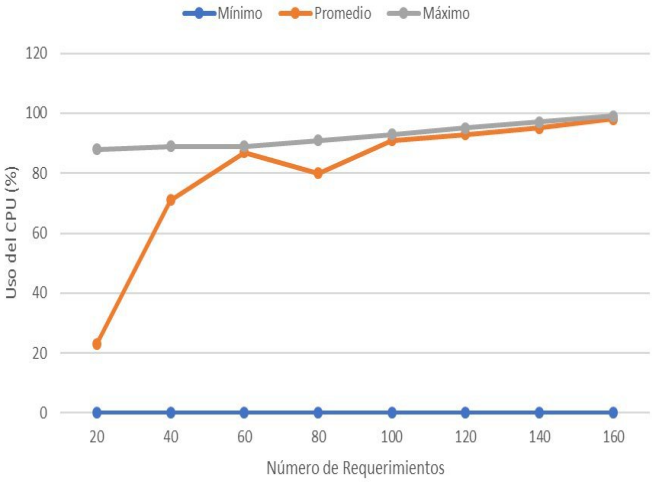

Figura 4: Porcentaje del Uso del CPU para el Set de Pruebas #1

La Figura 5 representa el tiempo de respuesta total para que el sistema pudiese procesar las peticiones. Se puede observar que el tiempo total aumenta en promedio entre 60 y 80 segundos, al incrementar el número de peticiones en 20. Eso es, tenemos un tiempo de procesamiento para el reconocimiento facial del orden de 3 a 4 segundos, cuando el número de imágenes en el sistema es fijo e igual a 50, y el número de peticiones simultáneas es significativo (entre 20 y 160 peticiones). Es importante mencionar que una petición de reconocimiento aislada o con un sistema sin carga, cuando se tienen 50 imágenes en el sistema, se tarda aproximadamente 2 segundos. Obtener un tiempo de respuesta del orden de 3 o 4 segundos cuando el sistema esté estresado (entre 20 y 160 peticiones), es una muestra que escala bien, dentro de este rango de peticiones en paralelo.

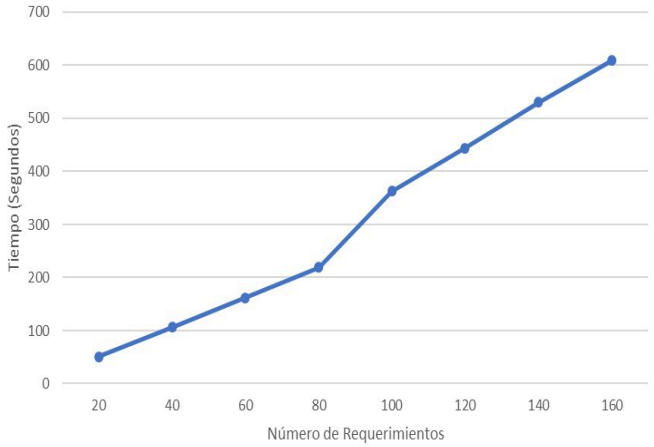

Figura 5: Tiempo de Respuesta Total para Procesar las Peticiones del Set de Pruebas #1

En el segundo set de pruebas realizadas, se mantuvo fijo el número de peticiones y se varió el número de imágenes a procesar, como se puede ver en la Tabla II.

Tabla II: Datos Asociados al Set de Pruebas #2

Peticiones	40	40	40	40	40	40	40	40
Imágenes	20	40	60	80	100	120	140	160

En la Figura 6, se observa el uso de la memoria RAM en la máquina virtual para cada experimento del segundo set de pruebas. Se puede ver que las tres curvas (mínimo, promedio y máximo de uso de memoria RAM) son muy similares entre sí, y el aumento del consumo de memoria es lento. Eso es, el número de imágenes en la base de datos afecta ligeramente el uso de la memoria RAM. Sin embargo, en comparación con el set de pruebas #1 (ver Figura 2), el uso de la memoria RAM es menor en este caso.

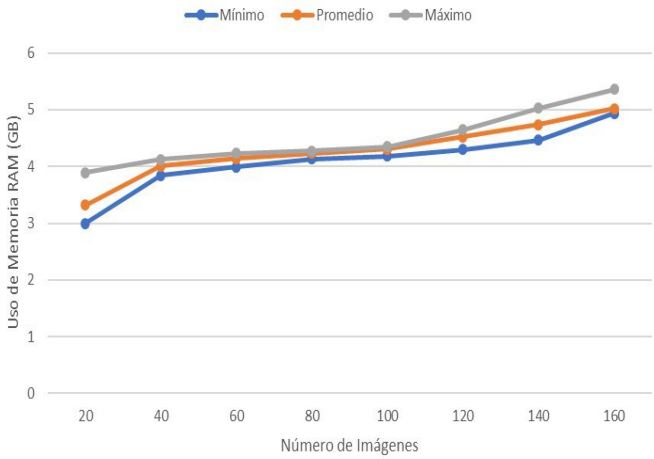

Figura 6: Uso de Memoria RAM para el Set de Pruebas #2

En la Figura 7, se observa la carga promedio de la máquina virtual con este nuevo set de pruebas. Se evidencia que la mayor carga registrada fue en la última prueba (160 imágenes) con una carga de 47 procesos para una máquina virtual con 8 núcleos de procesamiento, lo que implica que en el peor de los casos, hubo 39 procesos encoladas en un momento dado.

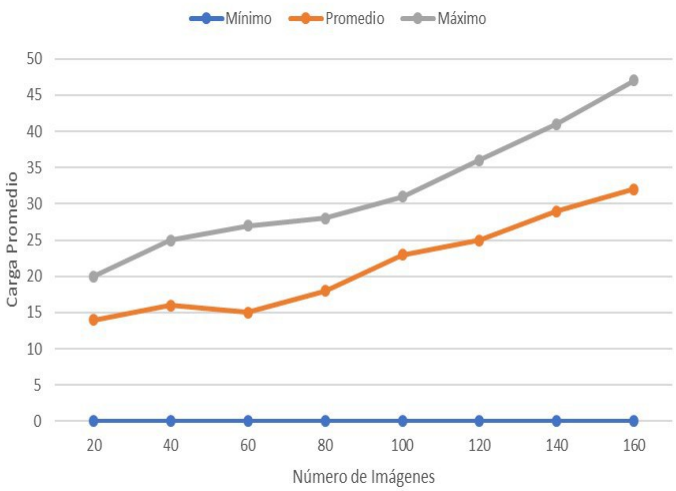

Figura 7: Carga Promedio para el Set de Pruebas #2

En cuanto al uso del CPU, al igual que en el set de prueba anterior (ver Figura 4), en la Figura 8, se evidencia que a medida que aumenta el número de imágenes, también aumenta el uso del CPU. Con la curva promedio, se puede observar que después de 120 imágenes, el CPU tiene un uso de por lo menos 80%, que se podría considerarse como la carga máxima a la cual se debería someter esta máquina virtual.

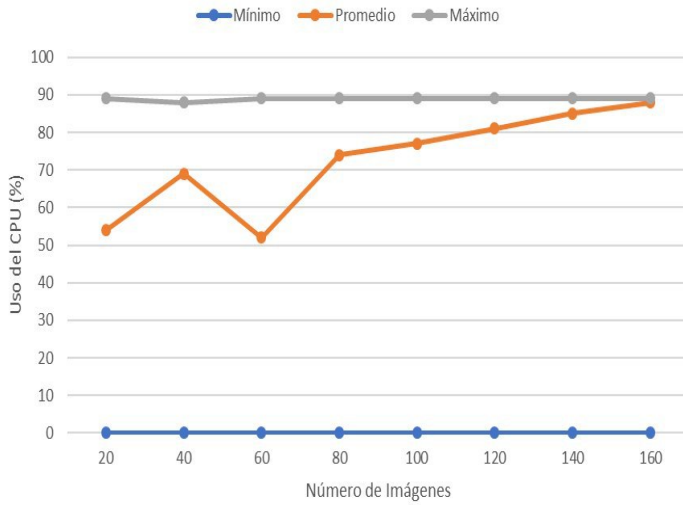

Figura 8: Porcentaje del Uso del CPU para el Set de Pruebas #2

El tiempo de respuesta total para atender todas las peticiones del segundo set de pruebas se muestra en la Figura 9. El menor resultado se obtiene para 20 imágenes y corresponde a un tiempo de respuesta promedio de 50/40 = 1.25 segundos. El mayor resultado se obtiene para 160 imágenes y corresponde a un tiempo de respuesta promedio de 430/40 = 10.75 segundos. Es importante mencionar que, para el caso de 160 imágenes almacenadas en el sistema, un reconocimiento facial aislado se tarda aproximadamente 4.50 segundos. Eso es, el sistema propuesto sigue escalando bien hasta un cierto punto.

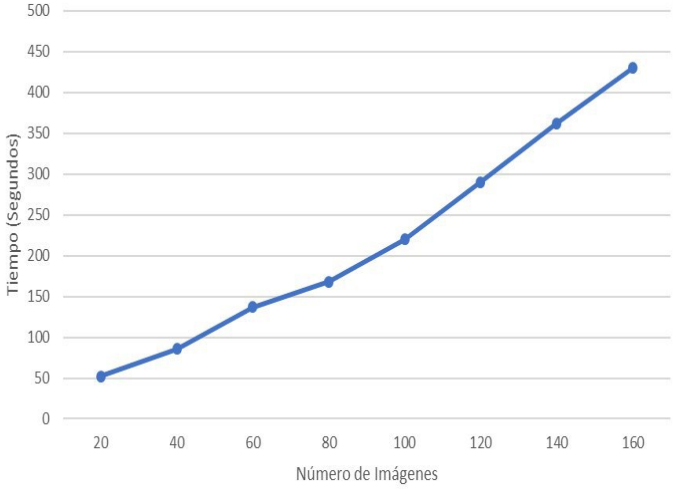

Figura 9: Tiempo de Repuesta las Peticiones del Set de Pruebas #2

V. Conclusiones

Con el constante crecimiento que han tenido las tecnologías involucradas en los sistemas de video vigilancia y el gran interés de llevar estos sistemas hasta los hogares, la comunidad se ha visto en la necesidad de desarrollar softwares de administración de sistemas de seguridad con cámaras IP que sean más adaptados a las necesidades de los usuarios finales. Por ejemplo, hoy en día, se necesitan soluciones de bajo costos con sistemas de administración que provean interfaces fáciles de utilizar, o que incluyan tecnologías como el reconocimiento facial para hacer más preciso el proceso de resguardo y vigilancia. Por lo antes dicho, en este trabajo de investigación, se evaluaron varios sistemas de administración de cámaras IP de código abierto: iSpy, Bluecherry, Shinobi, MotionEyes y ZoneMinder. Como resultado, fue elegido el software de administración de cámaras IP conocido como ZoneMinder [6][7] por ser uno de los más utilizados, y por su amplia documentación para desarrolladores que permitió la integración de otras tecnologías como lo es el reconocimiento facial.

Se desarrolló un API RESTful en lenguaje Python utilizando el framework Django [8-10] que permite a través de un conjunto de servicios y recursos, realizar todo el proceso de reconocimiento facial, desde llevar a cabo el registro de un usuario, hasta realizar un reconocimiento facial basado en una imagen dada. Este servicio está relacionado con un módulo que fue desarrollado directamente en el sistema de administración de cámaras IP de ZoneMinder, donde a través de un conjunto de interfaces web se pueden utilizar todos los recursos desarrollados en el API.

De igual modo, se realizaron un conjunto de refactorizaciones en las interfaces web del software de administración de cámaras IP de ZoneMinder, con la premisa de proporcionarle a los usuarios finales una mejor experiencia de uso a la hora de utilizar el sistema.

Con la intención de comprobar y validar la funcionalidad y el rendimiento de la solución, se realizaron un conjunto de pruebas. En la prueba de funcionalidad se comprobó el correcto comportamiento del módulo de reconocimiento facial y se verificó que las modificaciones realizadas a las interfaces no afectan el correcto funcionamiento del sistema. En las pruebas de rendimiento, con la herramienta JMeter [39-42], se realizaron

dos sets de pruebas al servicio de reconocimiento facial y se evaluó el uso de los recursos. Basado en estas pruebas de estrés, se puede concluir que el sistema sería óptimo para hogares y entidades pequeñas, con el uso de un computador con capacidades medianas, y sin GPU (Graphic Processing Unit). Para un sitio de tamaño mediano, se recomendaría contar con un GPU para el procesamiento digital de las imágenes, ya que utilizando únicamente el CPU, se llega a los límites del sistema en cuanto a uso de los recursos y al tiempo de respuesta de las peticiones de reconocimiento facial.

VI. Trabajos Futuros

Como posibles trabajos futuros, los autores están interesados en:

- Realizar una aplicación móvil que se comunique con el servidor de ZoneMinder y pueda proveer las mismas funcionalidades, pero desde el dispositivo móvil.
- Desarrollar un módulo de reconocimiento facial en el sistema de ZoneMinder bajo el lenguaje PHP, de tal forma que este sea propio del servicio, sin necesidad de contar con un API externa.
- Extender el módulo de reconocimiento facial para que también pueda realizar el reconocimiento de placas en automóviles.

Referencias

[1] J. Romanowich, D. Chin, and T. Lento, *Smart Video Security Handbook: A Practical Guide for Catching Intruders Before They Act*, Video Valley Press, 1 edition, October 2015.

[2] F. Nilsson, *Intelligent Network Video: Understanding Modern Video Surveillance Systems*, CRC Press, 2 edition, December 2016.

[3] N. Dey, A. Ashour, and S. Acharjee, *Applied Video Processing in Surveillance and Monitoring Systems (Advances in Multimedia and Interactive Technologies)*, IGI Global, 1st edition, October 2016.

[4] M. H. Kolekar, *Intelligent Video Surveillance Systems: An Algorithmic Approach*, Chapman and Hall/CRC, 1st edition, July 2018.

[5] E. Robins, *Recent Advances in Video Surveillance*, Clanrye International, February 2015.

[6] ZoneMinder, A Full-featured, Open Source, State-of-the-art Video Surveillance Software System, https://www.zoneminder.com.

[7] A. A. Deshmukh, A. D. Mihovska, and R. Prasad, *ZoneMinder as Software as a Service and Load Balancing of Video Surveillance Requests*, in procceedings of the 2012 ATSMA Networking and Electronic Commerce Research Conference (NAEC 2012), Riva del Garda, Italy, October 2012.

[8] A. Mele, *Django 3 by Example: Build Powerful and Reliable Python Web Applications from Scratch*, Packt Publishing, March 2020.

[9] A. Bendoraitis and J. Kronika, *Django 3 Web Development Cookbook: Actionable Solutions to Common Problems in Python Web Development*, Packt Publishing, 4th edition, March 2020.

[10] G. C. Hillar, *Django RESTful Web Services: The Easiest Way to Build Python RESTful APIs and Web Services with Django*, Packt Publishing, January 2018.

[11] Xeoma, A Video Surveillance Bestseller, https://felenasoft.com/xeoma/en.

[12] Kerberos.io, Video Surveillance Software for Everyone, http://www.kerberos.io.

[13] Milestone Systems, *XProtect*, https://www.milestonesys.com/solutions/platform/video-management-software.

[14] NCH Software, *EyeLine Video Surveillance Software*, https://www.nchsoftware.com/surveillance.

[15] ContaCam - *Video Surveillance Software*, https://www.contaware.com/contacam.html.

[16] Mobile Video Solutions Inc., *Ivideon: Smart Video Surveillance*, https://www.ivideon.com.

[17] Blue Iris Software, *Blue Iris*, https://blueirissoftware.com.

[18] Luxriot, *Evo S*, https://www.luxriot.com/product/video_surveillance/luxriot-evo-s.

[19] Moonware Studios, *webcamXP*, http://www.webcamxp.com/home.aspx.

[20] Hanwha Techwin Europe, *SmartViewer*, https://www.hanwha-security.eu/business-security-products/smart-viewer.

[21] Camcloud, *Cloud Video Surveillance Made Simple*, https://www.camcloud.com.

[22] ONVIF: Open Network Video Interface Forum, https://www.onvif.org.

[23] B. Burns, J. Beda, and K. Hightower, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*, O'Reilly Media, 2nd edition, October 2019.

[24] M. Karslioglu, *Kubernetes - A Complete DevOps Cookbook: Build and Manage your Applications, Orchestrate Containers, and Deploy Cloud-Native Services*, Packt Publishing, March 2020.

[25] M. D'Oliveiro, *The Streaming Media Guide: How to Successfully Integrate Streaming Media into your Communications Strategy*, 1st edition, Routledge, June 2019.

[26] H. Schulzrinne, A. Rao, and R. Lanphier, *Real Time Streaming Protocol (RTSP)*, RFC 2326, April 1998.

[27] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, *Real-Time Streaming Protocol Version 2.0*, RFC 7826, December 2016.

[28] iSpy, *iSpy User Guide*, https://www.ispyconnect.com/userguide.aspx.

[29] Bluecherry, *Linux Video Surveillance Software*, https://www.bluecherrydvr.com.

[30] Shinobi, *The Open Source CCTV Solution*, https://moeiscool.github.io/Shinobi.

[31] MotionEyeOS, *A Video Surveillance OS for Single-Board Computers,* https://github.com/ccrisan/motioneyeos/wiki.

[32] G. Lazar and R. Penea, *Mastering Qt 5: Create Stunning Cross-platform Applications using C++ with Qt Widgets and QML with Qt Quick*, Packt Publishing, 2nd edition, August 2018.

[33] J. Riselvato, *FFmpeg For Beginners: Edit Audio and Video Like a Pro for Youtube and Social Media*, Independently published, April 2020.

[34] Buildroot, *Making Embedded Linux Easy*, https://buildroot.org.

[35] C. Simmonds, *Mastering Embedded Linux Programming: Unleash the Full Potential of Embedded Linux*, 2nd edition, Packt Publishing, June 2017.

[36] D. Manchón Vizuete, *Instant Buildroot*, Packt Publishing, September 2013.

[37] M. Eusebio y E. Mello, *DroidMinder – Monitoramento de Câmeras de Vigilância Através de um Telefone Celular Android*, Instituto Federal de Santa Catarina (IFSC), São Jose, Brazil, 2010.

[38] R. Cucchiara, A. Prati y R. Besan, *Designing Video Surveillance Systems as Services*, Imagelab Lab, Information Engineering Department of University of Modena and Reggio Emilia, Italia, 2010.

[39] G. Blokdyk, *Apache JMeter: A Complete Guide*, 5STARCooks, July 2018.

[40] A. Rodrigues, B. Demion, P. Mouawad, *Master Apache JMeter - From Load Testing to DevOps: Master Performance Testing with JMeter*, Packt Publishing, August 2019.

[41] B. Erinle, *Performance Testing with JMeter 3: Enhance the Performance of your Web Application*, Packt Publishing, 3rd edition, July 2017.

[42] K. Rungta, *Learn JMeter in 1 Day: Definitive Guide to Learn JMeter for Beginners*, Independently published, May 2017.

[43] W. Ahmed, *Mastering Proxmox: Build Virtualized Environments using the Proxmox VE Hypervisor*, Packt Publishing, 3rd edition, November 2017.

[44] R. Goldman, *Learning Proxmox VE*, Packt Publishing, March 2016.

[45] W. Ahmed, *Proxmox Cookbook*, Packt Publishing, August 2015.

[46] Proxmox VE, An Open-Source Virtualization Platform: Compute, Network and Storage in a Single Solution, https://proxmox.com/en/proxmox-ve.

[47] S. Hyderkhan, *Performance of TCP in a KVM Virtualized Environment*, Lambert Academic Publishing (LAP), January 2019.

[48] Y.-J. Huang, H.-H. Wu, Y.-C. Chung, and W.-C. Hsu, *Building a KVM-based Hypervisor for a Heterogeneous System Architecture Compliant System*, in proceedings of the 12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Atlanta Georgia, USA, April 2016.

[49] S. Kumaran, *Practical LXC and LXD: Linux Containers for Virtualization and Orchestration*, Apress, 1st edition, September 2017.

[50] Grafana: The Open Observability Platform, https://www.grafana.com.