VNIVERSITAT DE VALÈNCIA

Aproximaciones en la preparación de contenido de vídeo para la transmisión de vídeo bajo demanda (VOD) con DASH

TESIS DOCTORAL

Wilmer Ariel Moina Rivera

Dirigida por:

Prof. Juan Gutiérrez Aguado (UV)
Prof. Miguel García Pineda (UV)

Programa de Doctorado en Tecnologías de la Información, Comunicaciones y Computación

Universitat de València

Julio 2023

Aproximaciones en la preparación de contenido de vídeo para la transmisión de vídeo bajo demanda (VOD) con DASH

Memoria que presenta para optar el título de

Doctor en Tecnologías de la Información, Comunicaciones y Computación

Por:

Wilmer Ariel Moina Rivera

Dirigida por:

Prof. Juan Gutiérrez Aguado (UV)
Prof. Miguel García Pineda (UV)

Programa de Doctorado en Tecnologías de la Información, Comunicaciones y Computación

Universitat de València

Julio 2023

A tus ojos verdes
y
a tí, lector carísimo

Agradecimientos

A todos los que la presente vieren y entendieren.

Inicio de las Leyes Orgánicas. Juan Carlos I

Antes que nada, deseo rendir homenaje a las dos almas que me han dado vida y sustento — Florita y Roldan —. A ella, en particular, por su amor incesante, por ser el faro en la tormenta y la roca inamovible en el remolino de este viaje académico. Su aliento constante y su apoyo incondicional han sido el viento que ha impulsado mis velas, dispersando las nubes de incertidumbre y guiándome hacia el puerto seguro de mi destino.

A mis tutores, les extiendo mi más sincero agradecimiento — Juan y Miguel —. Han sido los sabios fareros que han iluminado mi camino con su guía, depositando en mí una confianza que desde el primer día ha sido el cimiento sobre el que se ha erigido este trabajo. Su dedicación infatigable y su paciencia inagotable han sido las llaves que han abierto las puertas a la culminación de esta tesis.

A mis amigos, fieles compañeros en esta odisea, debo una deuda incalculable. Han sido el alivio en los momentos de tensión, el apoyo en los momentos de debilidad y la alegría en los momentos de victoria. Sin su amistad incondicional, este viaje habría sido un camino mucho más solitario.

Finalmente, quiero expresar mi gratitud a todos los demás miembros de mi familia y a todas las personas que de una forma u otra han contribuido a dar vida a esta tesis. A todos ustedes, les ofrezco mi más profundo agradecimiento. Han sido las manos invisibles que han ayudado a levantar este gran logro. Gracias por hacer posible este sueño.

No puedo concluir sin expresar mi gratitud a la persona que ha enfrentado cada obstáculo y desafío con resiliencia y determinación: a ti, Wilmer. Esta tesis es la materialización de sueños cultivados con esfuerzo y dedicación. Es un monumento a la persistencia y estoy orgulloso de la sombra que proyecta.

Resumen

Desocupado lector, sin juramento me podrás creer que quisiera que este libro [...] fuera el más hermoso, el más gallardo y más discreto que pudiera imaginarse.

Miguel de Cervantes, Don Quijote de la Mancha

El consumo de contenido multimedia a través de Internet, especialmente el vídeo, está experimentado un crecimiento constante, convirtiéndose en una actividad cotidiana entre individuos de todo el mundo. En este contexto, en los últimos años se han desarrollado numerosos estudios enfocados en la preparación, distribución y transmisión de contenido multimedia, especialmente en el ámbito del vídeo bajo demanda (VoD). Esta tesis propone diferentes contribuciones en el campo de la codificación de vídeo para VoD que será transmitido usando el estándar Dynamic Adaptive Streaming over HTTP (DASH). El objetivo es encontrar un equilibrio entre el uso eficiente de recursos computacionales y la garantía de ofrecer una calidad experiencia (QoE) alta para el espectador final.

Como punto de partida, se ofrece un estudio exhaustivo sobre investigaciones relacionadas con técnicas de codificación y transcodificación de vídeo en la nube, enfocándose especialmente en la evolución del *streaming* y la relevancia del proceso de codificación. Además, se examinan las propuestas en función del tipo de virtualización y modalidades de entrega de contenido.

Se desarrollan dos enfoques de codificación adaptativa basada en la calidad, con el objetivo de ajustar la calidad de toda la secuencia de vídeo a un nivel deseado. Los resultados indican que las soluciones propuestas pueden

reducir el tamaño del vídeo manteniendo la misma calidad a lo largo de todos los segmentos del vídeo.

Además, se propone una solución de codificación basada en escenas y se analiza el impacto de utilizar vídeo a baja resolución (downscaling) para detectar escenas en términos de tiempo, calidad y tamaño. Los resultados muestran que se reduce el tiempo total de codificación, el consumo de recursos computacionales y el tamaño del vídeo codificado.

La investigación también presenta una arquitectura que paraleliza los trabajos involucrados en la preparación de contenido DASH utilizando el paradigma FaaS (Function-as-a-Service), en una plataforma *serverless*. Se prueba esta arquitectura con tres funciones encapsuladas en contenedores, para codificar y analizar la calidad de los vídeos, obteniendo resultados prometedores en términos de escalabilidad y distribución de trabajos.

Finalmente, se crea una herramienta llamada VQMTK, que integra 14 métricas de calidad de vídeo en un contenedor con Docker, facilitando la evaluación de la calidad del vídeo en diversos entornos. Esta herramienta puede ser de gran utilidad en el ámbito de la codificación de vídeo, en la generación de conjuntos de datos para entrenar redes neuronales profundas y en entornos científicos como educativos.

En resumen, la tesis ofrece soluciones y herramientas innovadoras para mejorar la eficiencia y la calidad en la preparación y transmisión de contenido multimedia en la nube, proporcionando una base sólida para futuras investigaciones y desarrollos en este campo que está en constante evolución.

Abstract

Dear idle reader, without an oath you can believe me that I would like this book [...] to be the most beautiful, the most gallant, and the most discreet that could be imagined.

Miguel de Cervantes, Don Quijote de la Mancha

The consumption of multimedia content over the Internet, especially video, is growing steadily, becoming a daily activity among people around the world. In this context, several studies have been developed in recent years focused on the preparation, distribution, and transmission of multimedia content, especially in the field of video on demand (VoD). This thesis proposes different contributions in the field of video coding for transmission in VoD scenarios using Dynamic Adaptive Streaming over HTTP (DASH) standard. The goal is to find a balance between the efficient use of computational resources and the guarantee of delivering a high-quality experience (QoE) for the end viewer.

As a starting point, a comprehensive survey on research related to video encoding and transcoding techniques in the cloud is provided, focusing especially on the evolution of streaming and the relevance of the encoding process. In addition, proposals are examined as a function of the type of virtualization and content delivery modalities.

Two quality-based adaptive coding approaches are developed with the objective of adjusting the quality of the entire video sequence to a desired level. The results indicate that the proposed solutions can reduce the video size while maintaining the same quality throughout all video segments.

In addition, a scene-based coding solution is proposed and the impact of using downscaling video to detect scenes in terms of time, quality and size is analyzed. The results show that the required encoding time, computational resource consumption and the size of the encoded video are reduced.

The research also presents an architecture that parallelizes the jobs involved in content preparation using the FaaS (Function-as-a-Service) paradigm, on a serverless platform. This architecture is tested with three functions encapsulated in containers, to encode and analyze the quality of the videos, obtaining promising results in terms of scalability and job distribution.

Finally, a tool called VQMTK is developed, which integrates 14 video quality metrics in a container with Docker, facilitating the evaluation of video quality in various environments. This tool can be of great use in the field of video coding, in the generation of datasets to train deep neural networks, and in scientific environments such as educational.

In summary, the thesis offers innovative solutions and tools to improve efficiency and quality in the preparation and transmission of multimedia content in the cloud, providing a solid foundation for future research and development in this constantly evolving field.

Resum

Estimat lector desocupat, sense jurament em podràs creure que mágradaria que aquest llibre [...] fos el més bell, el més galant i el més discret que es pogués imaginar.

Miguel de Cervantes, Don Quijote de la Mancha

El consum de contingut multimèdia a través d'Internet, està experimentant un creixement constant, convertint-se en una activitat quotidiana entre individus de tot el món. En aquest context, en els últims anys s'han desenvolupat nombrosos estudis enfocats en la preparació, distribució i transmissió de contingut multimèdia, especialment en l'àmbit del vídeo sota demanda (VoD). Aquesta tesi proposa diferents aportacions en el camp de la codificació de vídeo per a la transmissió en escenaris de VoD mitjançant l'estàndard Dynamic Adaptive Streaming over HTTP (DASH). L'objectiu és trobar un equilibri entre l'ús eficient de recursos computacionals i la garantia d'oferir una qualitat d'experiència (QoE) alta per a l'espectador final.

Com a punt de partida, s'ofereix un estudi exhaustiu sobre investigacions relacionades amb tècniques de codificació i transcodificació de vídeo en el núvol, centrant-se especialment en l'evolució del streaming i la rellevància del procés de codificació. A més, s'examinen les propostes en funció del tipus de virtualització i modalitats de lliurament de contingut.

Es desenvolupen dos enfocaments de codificació adaptativa basats en la qualitat, amb l'objectiu d'ajustar la qualitat de tota la seqüència de vídeo a un nivell desitjat. Els resultats indiquen que les solucions proposades poden reduir la grandària del vídeo mantenint la mateixa qualitat al llarg de tots

els segments del vídeo.

A més, es proposa una solució de codificació basada en escenes i s'analitza l'impacte d'utilitzar vídeo a baixa resolució (downscaling) per detectar escenes en termes de temps, qualitat i mida. Els resultats mostren que es redueix el temps total de codificació, el consum de recursos computacionals i la grandària del vídeo codificat.

La investigació també presenta una arquitectura que paral · lelitza les tasques involucrades en la preparació de contingut fent ús del paradigma FaaS (Function-as-a-Service), en una plataforma serverless. Es prova aquesta arquitectura amb tres funcions encapsulades en contenidors, per codificar i analitzar la qualitat dels vídeos, obtenint resultats prometedors en termes d'escalabilitat i distribució de tasques.

Finalment, es crea una eina anomenada VQMTK, que integra 14 mètriques de qualitat de vídeo en un contenidor amb Docker, facilitant l'avaluació de la qualitat del vídeo en diversos entorns. Aquesta eina pot ser de gran utilitat en l'àmbit de la codificació de vídeo, en la generació de conjunts de dades per entrenar xarxes neuronals profundes i en entorns científics com educatius.

En resum, la tesi ofereix solucions i eines innovadores per millorar l'eficiència i la qualitat en la preparació i transmissió de contingut multimèdia en el núvol, proporcionant una base sòlida per a futures investigacions i desenvolupaments en aquest camp que està en constant evolució.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

El experimento es el único medio de conocimiento a nuestro alcance. Todo lo demás es poesía, imaginación.

Max Planck

Resumen:

A lo largo de este capítulo, se muestra una descripción general del proceso de preparación de contenido multimedia para la transmisión de vídeo a través de *HTTP Adaptive Streaming* (también conocido como HAS) en las condiciones de un escenario de vídeo bajo demanda (en inglés Video-On-Demand, VOD). Además, se presenta una lista de los objetivos que han guiado el desarrollo de esta tesis doctoral, una descripción breve de los trabajos realizados previamente por el grupo de investigación en esta área y un esquema sobre la estructura de este trabajo.

1.1. Preparación de contenido para la transmisión de vídeo bajo demanda

En todo el mundo, Internet continúa transformando la manera en que las personas nos comunicamos unas con otras, gestionamos nuestra vida cotidiana e intercambiamos información (Ismagilova et al., 2017). Como resultado, el consumo de contenido multimedia (imagen, audio, vídeo, etc.) ha cambiado drásticamente en los últimos años, desde el uso de la televisión (TV) tradicional —En casa, era necesario disponer de una videocasetera y una película en formato físico para disfrutar de los estrenos cinematográficos más recientes—, el ordenador de escritorio hasta el teléfono móvil. La innovación de estas tecnologías y la proliferación de los nuevos dispositivos convergen hoy en este punto, el uso de Internet, tanto para consumir o suministrar contenido multimedia a la red. Según las previsiones de (Cisco, 2020), en el 2023 habrá un total de 3.6 dispositivos conectados a la red (29.3 billones de dispositivos), lo que dará acceso al 66 % de la población mundial a Internet. En Abril del 2022 ya había más de cinco mil millones de usuarios activos en Internet en todo el mundo, lo que suponía el 63.1 % de la población mundial. De este total, 4.7 mil millones de personas, o el 59 % de la población, utiliza las redes sociales; gran parte del contenido multimedia que se comparte en estas redes es vídeo (Statista, 2022).

La participación en los vídeos y el entretenimiento en línea sigue siendo una de las actividades más populares entre los usuarios (Dwivedi et al., 2021). Además, el consumo de este tipo de contenido, ya sea vídeo en directo o vídeo bajo demanda, va en aumento y no muestra signos de desaceleración.

A día de hoy, más del 85 % del tráfico de Internet es contenido de vídeo, debido al uso masivo de las plataformas que ofrecen servicios de transmisión (en inglés *streaming*) como: *TikTok*[1], *Twitch*[2], *etc.*, las ya conocidas *Meta*[3] (p. ej., *Facebook* e *Instagram)*, *YouTube*[4], *Netflix*[5], *HBO Max*[6], las orientadas a la enseñanza-aprendizaje en linea o e-Learning (p. ej., *Coursera*[7] y *Udemy*[8])

[1] https://www.tiktok.com/
[2] https://www.twitch.tv/
[3] https://www.meta.com/
[4] https://www.youtube.com/
[5] https://www.netflix.com/
[6] https://www.hbomax.com/
[7] https://www.coursera.org/
[8] https://www.udemy.com/

(Suciu et al., 2017), las de videovigilancia orientados a la seguridad (Le Dao et al., 2018), entre otras. Por ejemplo, plataformas como YouTube o Netflix generaron el 26 % del tráfico global durante las órdenes de confinamiento de la pandemia mundial de COVID-19 (Sandvine, 2020).

Otro factor que contribuye a este volumen de tráfico, según Cisco (2020), se debe a que los usuarios consumen cada vez más contenido a alta resolución[9] como 4K/Ultra HD (UHD), e incluso 8K. Según el mismo estudio, cada persona pasa una media de 170 minutos al día en Internet. Además, el nuevo comportamiento y la satisfacción que experimentan los usuarios al consumir contenidos —especialmente los que se ofrecen bajo demanda— es otro factor a tener en cuenta (Pereira y Tam, 2021).

Según la clasificación realizada en (Li et al., 2020a), un escenario de transmisión de vídeo puede ser de tres tipos: transmisión bajo demanda, transmisión en directo (tiempo real o *Live*) y transmisión *Live-to-VOD*. En la transmisión VOD también conocida como *Over-The-Top* (OTT), el usuario puede consumir el contenido de vídeo cuando lo desee. En la transmisión en directo, el usuario podrá acceder al contenido cuando esté activa la transmisión. Mientras que, en la transmisión *Live-to-VOD*, una vez finalizada la conexión, el contenido producido durante la transmisión en vivo se pondrá a disposición de los usuarios como vídeo bajo demanda.

Como se muestra en la Figura 1.1 a un escenario de transmisión de vídeo, pueden acceder simultáneamente y de forma ubicua[10] muchos usuarios con distintas configuraciones de tamaño de pantalla, resolución, potencia de procesamiento, ancho de banda de conexión a Internet (p. ej., banda ancha, Wi-Fi, LTE, 3G, 4G y 5G), etc. Debido a esta gran variedad, los sistemas de codificación de vídeo actuales necesitan preparar varias representaciones de un mismo vídeo.

Debido a la cantidad de usuarios en la red de Internet y a la variedad de dispositivos, las plataformas de *streaming* han ido adaptando sus sistemas para la codificación (preparación de contenido), la distribución y la transmisión de contenido de vídeo (Liu et al., 2020; Bhat et al., 2017; Bentaleb et al., 2019).

[9] Resolución de vídeo: Se refiere a la cantidad de píxeles que se muestran en la pantalla durante la reproducción de un vídeo. Cuanto mayor sea la resolución, más detallado será el vídeo.

[10] Ubicua en tecnología: Se dice de cuando podemos estar conectados a la red de Internet en todo momento, sin importar el lugar (Wikipedia, Ubicuo).

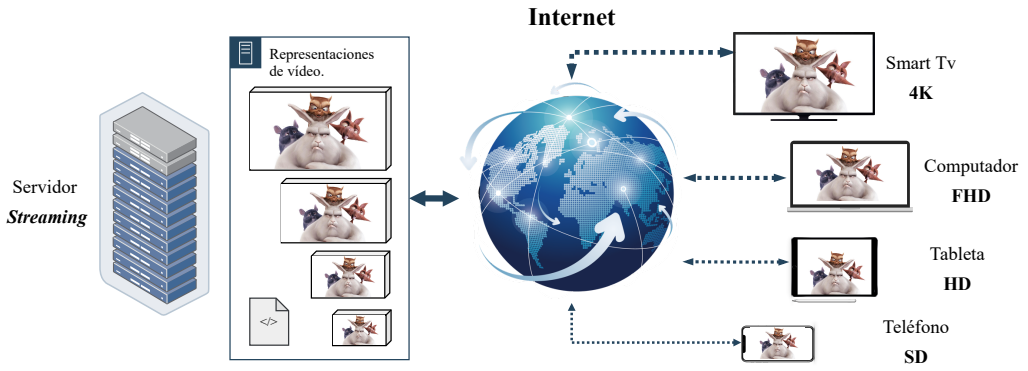


Figura 1.1: Escenario de transmisión de vídeo mediante tecnología HAS.

Actualmente, la mayoría de las plataformas utilizan el protocolo HTTP para ofrecer sus contenidos en *streaming*, implementando especificaciones tecnológicas basadas en HAS para mejorar la calidad de experiencia del usuario (QoE) (Seufert et al., 2015). En HAS, durante la fase codificación, un vídeo se divide en segmentos temporales disjuntos[11] (o *chunks*) pequeños, del orden de segundos, típicamente entre 2 y 10 segundos (Lederer, 2015), y cada segmento se procesa independientemente ajustando diferentes parámetros de codificación, para obtener como resultado varias representaciones de un mismo segmento de vídeo con distintos niveles de calidad. Mientras tanto, en el lado del cliente, cada reproductor ejecuta un algoritmo para seleccionar el segmento más adecuado en función de las diferentes condiciones de red con el fin de mitigar problemas como el retraso inicial o el estancamiento durante el proceso de reproducción (Bentaleb et al., 2019). Esto permite proporcionar el vídeo de la manera más eficaz y con la mayor calidad posible para cada usuario y dispositivo en particular.

Hay varias especificaciones del formato HAS, pero las más utilizadas son *HTTP Live Streaming* (HLS) (Apple, 2022) y *Dynamic Adaptive Streaming over HTTP* (DASH) (ISO Central Secretary, 2022). En una transmisión con DASH, el proceso de codificación puede ser tan importante como la selección de la representación más adecuada para un segmento en el lado del cliente.

En este primer proceso, los sistemas que brindan este tipo de servicio tienen que definir un método para procesar el contenido de vídeo, donde se suele considerar factores como:

[11] Segmento de vídeo: Conjunto contiguo de imágenes (o fotogramas) correspondientes a una secuencia de vídeo (ITU-T, 2016a).

- el tipo de segmentación (puede ser fija o variable),
- el tamaño del segmento (mínimo y máximo si es segmentación variable),
- los códecs a usar[12],
- los parámetros para la codificación (tasa de bits o *bitrate*[13], resolución, fotogramas por segundo o *fps*[14], *Constant Rate Factor* o CRF, etc.)

con el objetivo de alcanzar un compromiso entre la calidad y el espacio de almacenamiento del vídeo resultante.

Los parámetros utilizados a lo largo del proceso de codificación y en los pasos previos pueden tener una influencia significativa en el rendimiento del sistema en términos de uso de recursos (CPU, memoria RAM, energía, etc.) (Wei et al., 2017; Bross et al., 2021b; Zach y Slanina, 2018a; De Cock et al., 2016; Tseng et al., 2018).

Además, la elección de unos determinados parámetros de codificación puede orientar hacia un método de codificación en concreto, por ejemplo, en la codificación de *bitrate* constante (en inglés *Constant Bitrate Encoding* -CBR), dónde el códec procesa cada segmento de vídeo con el objetivo de mantener un valor de *bitrate* a lo largo de toda la secuencia de vídeo[15]. Plataformas como YouTube proporcionan un rango de valores de *bitrate* por resolución como referencia, estos valores de *bitrate* puede asegurar un cierto nivel de calidad[16]. En cuanto a la codificación variable (en inglés *Variable Bitrate Encoding* -VBR), uno de sus subtipos, es la calidad constante VBR

[12]Un códec de vídeo: Permite comprimir y descomprimir contenido de vídeo. Normalmente los algoritmos de compresión empleados conllevan una pérdida de información. Su finalidad es obtener un almacenamiento sustancialmente menor a la información de vídeo en bruto (en inglés RAW), manteniendo un compromiso de calidad (Wikipedia, Códec de video).

[13]En la transmisión de vídeo, la tasa de bits se refiere al número de bits que se transmiten o procesan en una unidad de tiempo. La tasa de bits se suele medir en bits por segundo (bit/s, bps).Por lo general, cuanto mayor sea la tasa de bits, mayor será la calidad del vídeo, pero también el tamaño del archivo.

[14]Fotogramas por segundo: Es la frecuencia (tasa) a la cual un dispositivo muestra imágenes llamadas fotogramas. Cuanto mayor sea la tasa de fotogramas, más suave será la reproducción del vídeo.

[15]Secuencia de vídeo: Flujo audiovisual compuesto por múltiples segmentos no superpuestos. (ITU-T, 2016a).

[16]YouTube recommended upload encoding settings `https://support.google.com/youtube/answer/1722171`

(en inglés *Quality VBR*, también conocida como *Constant Quality* -CQ), cuyo objetivo es mantener un nivel constante de calidad a lo largo de toda la secuencia de vídeo durante la codificación. El CRF es un parámetro que se utiliza para este fin; los valores más bajos de este parámetro dan como resultado segmentos codificados de mayor tamaño, pero de mayor calidad. A diferencia de la codificación CBR, el valor del *bitrate* variará entre segmentos, ya que el codificador busca mantener un nivel de calidad constante, considerando la complejidad del contenido. Algunos de los códecs que permiten estos métodos de codificación son x264/AVC[17], x265/HEVC[18], VP9[19], AV1[20], etc.

En la preparación del contenido de vídeo para una transmisión de VOD, se puede seleccionar cualquiera de los métodos mencionados. En los dos casos se obtendrá una variedad de representaciones del mismo vídeo, para ser servidos a través de una tecnología como DASH. Sin embargo, el tamaño de almacenamiento y el tiempo de procesamiento de un vídeo pueden variar entre los dos métodos, ya sea por la implementación propia del códec o por los pasos establecidos antes de la codificación.

Con CBR, es posible determinar si un vídeo se ajusta al valor objetivo de *bitrate* después de haber sido codificado mirando sus metadatos; sin embargo, con CQ es necesario realizar una evaluación del vídeo para determinar el valor de calidad obtenido, porque, el valor de CRF no se correlaciona directamente con un valor específico de calidad de vídeo, como podremos ver en los experimentos realizados en el Capítulo 3.

Para medir la calidad de un vídeo (o en inglés Video Quality Assesment, VQA), se pueden llevar a cabo evaluaciones subjetivas u objetivas de calidad. La evaluación subjetiva de calidad de vídeo, radica en conocer la opinión de numerosos sujetos[21] tras una experiencia de visualización del contenido codificado. Existen normas estandarizadas como la ITU-T P.913 (ITU-T, 2021) que se pueden seguir para definir las condiciones[22] de la evaluación subjetiva. De este tipo de evaluación por lo general se obtiene como resultado la puntuación media de opinión (en inglés Mean Opinion Score o MOS) que representa el promedio de las opiniones recabadas para un caso de uso en particular, sobre la "calidad percibida" para una persona.

[17] https://code.videolan.org/videolan/x264.git
[18] https://bitbucket.org/multicoreware/x265_git/wiki/Home
[19] https://chromium.googlesource.com/webm/libvpx
[20] https://aomedia.googlesource.com/aom
[21] Sujeto: Participante en un experimento subjetivo (ITU-T, 2016b).
[22] Condición: Uno de una serie de casos de uso que se evalúan en un experimento subjetivo; circuito de referencia hipotético (HRC) en experimentos de vídeo (ITU-T, 2016b).

Sin embargo, debido al coste de los recursos y la cantidad de contenido producido, sería inviable utilizar personas para evaluar el contenido de vídeo en el marco de una plataforma de servicios de VOD (p. ej., Netflix). Como alternativa se ha desarrollado la evaluación objetiva de calidad de vídeo, donde un algoritmo de medición de la calidad, tiene por objetivo predecir una experiencia subjetiva. Para realizar este tipo de evaluación la investigación y la industria han realizado varias propuestas de métricas de calidad de vídeo como las conocidas *PSNR*, *SSIM*, *MSSIM*, *VQM*, etc., y las introducidas en los últimos años como *VMAF*, *CAMBI* (ambas desarrolladas por el equipo de Netflix), etc.

Es importante tener en cuenta que la calidad del vídeo en una transmisión tiene un impacto directo en la experiencia de usuario. Además, la calidad del servicio (QoS) también puede afectar la QoE. La QoS se refiere a la calidad del rendimiento de la red y se mide a través de parámetros como el tiempo de retraso, la pérdida de paquetes y el jitter. Por lo tanto, es importante mantener altos niveles de calidad en ambos aspectos para garantizar una buena experiencia de usuario durante la transmisión (Wang et al., 2010; Chen et al., 2015).

Debido a la complejidad computacional requerida para trabajar con altas resoluciones (por ejemplo, 4K), cualquier proceso que involucre la codificación, transcodificación[23] o determinar la calidad de un vídeo puede suponer un tiempo de procesamiento prolongado. Por esta razón, estos procesos han sido acelerados mediante el uso de soluciones hardware como aceleradores por GPU (Wichtlhuber et al., 2016; Rodríguez-Sánchez et al., 2013) y plataformas multinúcleo (Asif et al., 2014). Si se requiere una mayor aceleración, se necesitan soluciones distribuidas, en las que la carga se reparta entre diversos nodos de procesamiento.

El Instituto Nacional de Estándares y Tecnología describe la computación en Nube como un modelo que permite el acceso ubicuo, práctico y bajo demanda a un conjunto compartido de recursos informáticos configurables (p. ej., redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados (escalabilidad) con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios (elasticidad) para servir a múltiples clientes utilizando un modelo multiarrendatario (Mell et al., 2011). Por tanto, la infraestructura basada en la Nube puede ser un componente crucial para almacenar, codificar y transcodificar contenido multimedia de forma fiable (Armbrust et al., 2010). Netflix, por ejemplo, emplea

[23]Transcodificación: El vídeo a procesar ya está codificado.

Amazon Web Services (AWS) para casi todas sus necesidades informáticas y de almacenamiento, incluidas las bases de datos, los análisis, los motores de recomendación, la transcodificación de vídeo y otras cientos de funciones (Netflix, 2022). Según el informe anual Bitmovin (2021), el uso de servicios de codificación en la Nube está en aumento en general, del 29 % en 2019 al 32 % en 2020. En 2021, el 33 % de los encuestados utilizaban la Nube para la codificación en directo y el 37 % para VOD. Esto da paso a seguir investigando y planteando soluciones para la codificación de vídeo basada en la Nube.

En cuanto a la entrega, una vez que el contenido de vídeo ya está preparado, los creadores de contenido utilizan las CDN (en inglés Content Delivery Network) para distribuir su contenido geográficamente, en una red cercana a sus usuarios, para garantizar un cierto nivel de calidad de servicio (Fan et al., 2018).

A la luz de esta premisa, la principal motivación de esta tesis doctoral es desarrollar un sistema que permita la preparación de contenido de vídeo para su posterior transmisión en un escenario de vídeo bajo demanda, utilizando la transmisión adaptativa con el formato DASH. Para ello, se estudió los beneficios que supondría integrar una métrica de calidad objetiva (como VMAF) como parte del método de codificación, con el propósito de obtener representaciones que se ajusten a un valor objetivo de calidad definida por el usuario. Se investigó, cómo el proceso de *downscaling* (o la reducción de la resolución) del vídeo fuente para obtener las escenas impactaría en el tiempo, la calidad y el tamaño del vídeo codificado en sus múltiples representaciones; esto se realizó para permitir una codificación basada en escenas (en lugar de usar segmentos de duración fija). Además, se examina el comportamiento de las funciones *serverless* (o funciones sin servidor) dirigidas por eventos sobre entornos *cloud* para codificar los segmentos de vídeo y, opcionalmente, calcular su calidad; esto es posible, porque, las tareas de codificación de los segmentos se pueden realizar en paralelo.

Finalmente, otra motivación importante es desarrollar una herramienta *open source* que incluya varias métricas de calidad de vídeo para realizar evaluaciones objetivas de calidad. Esta herramienta debe operar en entornos multiplataforma y estará disponible como una herramienta de código abierto para que las organizaciones de investigación y empresariales puedan usarla, modificarla, mejorarla y distribuirla.

A título personal: desarrollar una herramienta que permita agilizar varias tareas y luego ponerla a disposición de los demás, es otra forma de expresar

gratitud a la comunidad de investigación por todo el conocimiento compartido.

1.2. Objetivos de la investigación

En este subapartado, presentamos los siguientes objetivos y los pasos involucrados en el procedimiento para conseguir nuestra idea principal, que plantea un conjunto de aproximaciones para la preparación de contenido de vídeo para su posterior transmisión en un escenario de vídeo bajo demanda utilizando la transmisión adaptativa mediante el estándar DASH. Para desarrollar esta idea y mejorarla, debemos realizar las siguientes tareas:

1. Realizar un estudio exhaustivo sobre los trabajos existentes relacionados con la codificación de vídeo en la Nube y las arquitecturas MEC.
    a) Realizar un análisis de las propuestas relacionadas con la codificación de vídeo en la Nube de los últimos años y agruparlas según la tecnología utilizada en la Nube, y según la finalidad de cada propuesta.
    b) Presentar una breve reseña de las revisiones bibliográficas y los estudios pertinentes en este ámbito.
    c) Presentar un análisis de los retos a los que deberán enfrentarse los investigadores y la industria para ofrecer servicios de *streaming* de alta calidad.
2. Desarrollar e implementar aproximaciones de codificación de vídeo multiresolución basado en la calidad para escenarios de VOD con DASH.
    a) Diseñar e implementar un algoritmo prototipo (esquema I, véase la Sección 3.2) que, dada una métrica y un valor objetivo de calidad, proporcione los parámetros de codificación por segmento para generar un vídeo ajustado al valor de calidad objetivo.
    b) Desarrollar un proceso dinámico de codificación recursivo (esquema II, véase la Sección 3.3) para adaptar el factor de velocidad constante (o CRF) a cada segmento con el fin ajustar el vídeo a un valor objetivo de calidad proporcionado por el usuario.
    c) Implementar las soluciones sobre una arquitectura basada en contenedores.

d) Analizar los tiempos de codificación para validar la eficacia de cada sistema.

e) Validar el sistema con otras métricas de calidad para comprobar que la propuesta funciona correctamente.

f) Comparar la codificación adaptativa propuesta con la codificación fija de CRF, es decir, usando el mismo valor de CRF para todos los segmentos.

g) Desarrollar una página web demo que implemente un escenario de transmisión de vídeo bajo demanda con DASH para reproducir un ejemplo de las soluciones propuestas.

3. Estudiar el efecto del *downscaling* para la detección de escenas en el tiempo de codificación, la calidad y el tamaño de los vídeos codificados.

   a) Programar las funciones que realicen el proceso de *downscaling* del vídeo, la segmentación por escenas a partir del vídeo resultante, la codificación y la preparación del contenido para una transmisión con DASH.

   b) Implementar la solución en una arquitecta local basada en contenedores Docker.

   c) Monitorizar el uso de recursos hardware (CPU y memoria RAM) usados en la detección de escenas con/sin *downscaling* previo.

   d) Observar los efectos de la propuesta en el tiempo de codificación, la calidad y el tamaño de los vídeos codificados.

   e) Analizar la propuesta al codificar el contenido a distintas resoluciones y niveles de calidad.

   f) Desarrollar una página web de demostración donde se pueda reproducir un ejemplo de la solución propuesta.

4. Analizar e implementar funciones *serveless* dirigidas por eventos para codificar segmentos de vídeo y calcular, opcionalmente, la calidad de los vídeos codificados.

   a) Desarrollar las funciones *serveless* de codificación de vídeo (x264 y AV1) y de calidad de vídeo (VMAF) basadas en eventos y encapsularlas en contenedores Docker.

   b) Analizar el comportamiento de las funciones desarrolladas en una plataforma *serverless on-premise* con recursos limitados. Se han de realizar experimentos sobre escalabilidad y consumo de recursos.

   c) Analizar la solución propuesta para un escenario de vídeo bajo demanda con DASH, mostrando su rendimiento y comportamiento variando el número y tipo de réplicas.

   d) Desarrollar y analizar el *pipeline* (canales en inglés) de eventos para codificar segmentos de vídeo y obtener la calidad del vídeo codificado a partir de un único evento enviado por los usuarios.

5. Diseñar e implementar una herramienta multiplataforma que integre varias métricas para evaluar la calidad de vídeo.

   a) Describir las métricas de calidad de vídeo seleccionadas.

   b) Integrar la herramienta en una tecnología para su ejecución multiplataforma.

   c) Proveer a la herramienta de un entorno visual donde poder calcular la calidad y analizar los resultados mediante el uso de Jupyter Notebooks.

   d) Analizar el consumo de recursos de la herramienta (CPU y memoria RAM) por métrica de calidad.

1.3. Trabajos previos

Dentro del grupo de investigación, ya existen algunos trabajos previos donde se utilizan métricas de calidad objetiva para mantener y estimar una buena calidad de experiencia en términos de calidad de vídeo. Por ejemplo, en (Segura-Garcia et al., 2015; Garcia-Pineda et al., 2016) mis directores ya proponían analizar métricas de calidad de vídeo para predecir o estimar el MOS subjetivo basadas en técnicas de Análisis Factorial (AF) tanto con enfoques VQA de Referencia Completa (Full Reference – FR)[24] como Sin Referencia (No Reference – NR)[25] en la transmisión de vídeo en directo a través de servicios CMM (Cloud Mobile Media). En este trabajo, según el porcentaje de la varianza acumulada, conseguían una precisión en la estimación MOS de alrededor del 93 % en el peor caso con el enfoque FR y 78 % para NR. Sin embargo, mencionan que: no es suficiente para cumplir con los requisitos de una buena métrica de calidad de vídeo. Recomendaban analizar

[24]VQA de Referencia Completa: Requieren un acceso completo a la información del vídeo de referencia como del distorsionado para determinar la calidad.

[25]VQA Sin Referencia: La información del vídeo distorsionado es la única información utilizada para determinar la calidad.

el comportamiento de las variables relacionadas con la calidad de servicio, el flujo de bits y los parámetros de calidad de vídeo en un paso inicial —tal vez en este último inciso dieron luz a lo que hoy sería mi tema de tesis—, porque estas variables podrían ser eliminadas para reducir la complejidad computacional durante la transmisión, y/o incluir variables adicionales.

Por otro lado, con el objetivo de mantener la satisfacción del cliente en niveles *"buenos"*, en términos de MOS, en (García-Pineda et al., 2017) proponen una arquitectura adaptativa basada en redes definidas por software (Software Defined Networking –SDN) para operar en los servicios multimedia CMM. Estiman el MOS utilizando el método estadístico AF y luego el controlador SDN es capaz de programar diferentes acciones. Demuestran que las métricas propuestas son precisas con un PCC (Pearson Correlation Coefficient o R) por encima de 0.9 al compararlas con métricas de calidad de vídeo objetivas ampliamente utilizadas. Manteniendo el mismo hilo de encontrar una expresión holística para predecir y estimar el MOS, en (Garcia-Pineda et al., 2018) se procesan diferentes variables, con técnicas de estimación adicionales al AF como la Regresión Lineal Múltiple (Multinomial Linear Regression –MLR) y las Redes Neuronales Artificiales (Artificial Neural Networks –ANN), con los enfoques FR y NR. Se alcanzó un error de estimación promedio entre 0.46 y 15.94 % dependiendo de la técnica utilizada. En esta última aportación, como trabajo futuro se recomienda utilizar estos enfoques y técnicas en escenarios de transmisión de vídeo bajo demanda con el formato DASH.

En cuanto a la preparación del contenido para la transmisión de VOD, guiados por las características que ofrece la computación en la Nube, como la asignación de recursos bajo demanda, la *multi-tenancy*, la elasticidad, la escalabilidad, la tolerancia a fallos, entre otras, han diseñado, implementado y validado una arquitectura distribuida en la Nube para acelerar la codificación de vídeo en (Gutiérrez-Aguado et al., 2020a). Entre sus aportaciones esta la especificación de los nodos de codificación (o trabajadores) necesarios, la adaptación de la red y el almacenamiento, la especificación de los trabajos, el desarrollo de una aplicación distribuida basada en un *broker* de mensajes para asignar dinámicamente los trabajos de codificación, y la validación de la propuesta en una Nube privada. Para validar la arquitectura se ha evaluado la ejecución en paralelo de los códecs x265 y VP9, en términos de escalabilidad, carga de trabajo y distribución de trabajos, variando el número de nodos codificadores. Los resultados mostraron que, según el códec y el número de nodos codificadores, la codificación en paralelo puede reducir el tiempo necesario para la codificación hasta en un 90 %. En cuanto a la calidad, se

han codificado vídeos con varios tamaños de segmento (vídeo completo, 25, 50, 100 y 200 fotogramas), así como velocidades de bits (0.75 Mb, 1.5 Mb, 3 Mb y 4.5 Mb). La evaluación se llevó a cabo utilizando la métrica de calidad PSNR (del inglés Peak Signal-to-Noise Ratio), y se menciona que la solución propuesta brinda un nivel de calidad similar a la codificación secuencial.

En su siguiente aportación (Gutiérrez-Aguado et al., 2020b), utilizando una metodología similar al trabajo antes descrito, se evalúa la propuesta de forma más exhaustiva con los códecs x265, VP9, AV1. Los resultados denotan una disminución significativa del tiempo de ejecución en comparación con la codificación de vídeo completa (un solo nodo codificador o trabajador), que va desde el 61 % hasta cerca del 91 % cuando se utilizan de 3 a 15 nodos codificadores. Para evaluar la calidad de los vídeos codificados a diferentes velocidades de bits y tamaños de fragmentos, se emplearon métricas de calidad de FR como PSNR, MSSIM (Multi-Scale Structural Similarity) y VIF (Visual Information Fidelity). Observan que, el tamaño del fragmento tiene un impacto en la tasa de bits y la calidad del vídeo en todos los códecs. A medida que crece el tamaño del fragmento, la calidad y la tasa de bits tienden a los valores obtenidos al codificar el vídeo completo sin dividirlo. x265 es el códec más afectado por el cambio en el tamaño de los fragmentos y la tasa de bits, mientras que VP9 es el códec menos afectado. De acuerdo con los resultados obtenidos, dicen que 100 fotogramas (4 segundos) es el tamaño ideal para el segmento en este escenario ya que permite lograr una codificación rápida con una calidad comparable a la codificación de vídeo completo.

La descripción del trabajo futuro en esta última contribución sentaría las bases para el desarrollo de esta tesis doctoral. Se basaría en el conocimiento que el equipo de investigación ha adquirido de varias de sus publicaciones relacionadas con la codificación, transmisión y aseguramiento de la calidad de la experiencia, incluido el diseño y la implementación de arquitecturas de sistemas basadas en los modelos de la computación en la Nube, entre otras cosas. Ellos mencionan investigar nuevos esquemas de segmentación (división) de vídeo como el no uniforme y otras estrategias de programación de trabajos para mejorar el proceso de codificación mientras se mantiene el nivel de calidad visto por el usuario final. Aparte de eso, el introducir nuevas métricas de calidad de vídeo como VMAF (Video Multimethod Assessment Fusion) para que se tenga en cuenta la calidad perceptiva del vídeo durante la codificación.

Alternativamente, existen otros estudios relacionados con esta tesis, pero todos ellos han servido para investigar nuevos conceptos con respecto a la

propuesta de nuevas aproximaciones para la codificación de vídeo, presentadas en este trabajo. Por ello, se citarán estos trabajos en sus correspondientes capítulos para introducir las ideas o la solución de cada uno de ellos.

1.4. Listado de trabajos resultantes de esta tesis

A continuación, se muestran los trabajos que se han publicado o se encuentran en fase de revisión, a raíz de los avances que se han obtenido de esta tesis doctoral en el campo de la codificación de vídeo y su implementación sobre entornos FaaS (Function as a Service) desplegados en la nube.

- Moina-Rivera, W., Garcia-Pineda, M., Claver, J. M. y Gutiérrez-Aguado, J. Event-Driven Serverless Pipelines for Video Coding and Quality Metrics. Journal of Grid Computing, vol. 21(2), página 20, ISSN 1572-9184. 2023.

- Moina-Rivera, W., Garcia-Pineda, M., Gutiérrez-Aguado, J. y Alcaraz Calero, J. M. Cloud media video encoding: Review and challenges. Computer Science Review. 2023. En proceso de revisión con ID: COSREV-D-23-00107.

- Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Video Quality Metrics Toolkit: An Open Source Software to Assess Video Quality[26]. SoftwareX, vol. 23, página 101427. ISSN 2352-7110. 2023.

- Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Multiresolution quality-based video coding system for dash scenarios. En Proceedings of the 31st ACM Workshop on Network and Operating Systems Suppot for Digital Audio and Video, NOSSDAV 21. ISBN 9781450384353. 2021.

- Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Codificación de vídeo basada en vmaf para escenarios dash. En XV Jornadas de Ingeniería Telemática–JITEL 2021, vol. 15, páginas 116–121. 2021.

[26]Herramienta incluida en los repositorios de Video Quality Experts Group (VQEG), ver: https://vqeg.github.io/software-tools/quality%20analysis/vqmtk/

- Moina-Rivera, W., Gutiérrez-Aguado, J., Garcia-Pineda, M. y Claver, J. M. Improving dash encoding with scenes and downscaling techniques for vod streaming. En 2021 IEEE Global Communications Conference (GLOBECOM), páginas 1–6. 2021.

- Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. En ACM International Conference on Interactive Media Experiences (IMX) - Doctoral Consortium - Paper 3 - Wilmer Moina-Rivera. 2020.

1.5. Estructura de la tesis

Tras introducir las principales cuestiones que han motivado esta tesis y los principales objetivos que se persiguen, el resto de la memoria se organiza de la siguiente manera:

En el capítulo 2, mostramos un estudio detallado del estado del arte de varias aspectos (por ejemplo, escenarios de transmisión de vídeo, aseguramiento de la calidad, arquitecturas de Nube, etc.) relacionados con la codificación de vídeo, las arquitecturas de sistemas en la Nube y de borde, y las principales motivaciones para hacer converger estos dos paradigmas. Luego, se presenta un breve resumen de las revisiones de la literatura relacionados con cada capítulo de la tesis. Este capítulo se divide en seis secciones. En la sección 1, realizamos una introducción breve. En la sección 2, analizaremos la evolución del *streaming*, centrándonos especialmente en los últimos métodos de *streaming* y en la importancia de los procesos de codificación en los sistemas de *streaming* actuales. En sección 3, describimos la metodología que hemos utilizado para seleccionar los trabajos relacionados con el proceso de codificación o transcodificación de vídeo en infraestructuras en la Nube. En la sección 4, destacamos la importancia de los sistemas en la Nube en los entornos multimedia y detallamos la arquitectura de la infraestructura que se utiliza en la actualidad para proporcionar servicios de *streaming* en la Nube. En la sección 5, presentamos los trabajos relacionados con la preparación del contenido de vídeo. Finalmente, en la sección 6, exponemos las conclusiones derivadas del estudio realizado.

A continuación, en el capítulo 3, mostramos el desarrollo y la implementación de un sistema prototipo y un sistema de codificación multiresolución para DASH, en las propuestas se adapta el CRF a cada segmento con el fin ajustar el vídeo a un valor objetivo de calidad proporcionado por el usuario.

Este capítulo se organiza de la siguiente manera: la primera sección introduce la importancia de disponer de un sistema de codificación adecuado para los sistemas DASH. En la sección 2 y 3 se presenta el sistema prototipo (esquema I) y un sistema de codificación de vídeo multiresolución (esquema II) basados en la métrica de calidad VMAF, así como las diversas pruebas que validan las propuestas. Finalmente, en la sección 4 se presentan las conclusiones que se han obtenido del trabajo expuesto en este capítulo.

Seguidamente, en el capítulo 4 se estudia el efecto del *downscaling* para la detección de escenas en el tiempo, la calidad y el tamaño de los vídeos codificados. El capítulo está organizado como se indica a continuación. Una breve introducción del tema en la sección 1. A continuación, en la sección 2 se presenta nuestro sistema de codificación de vídeo basado en escenas y técnicas de *downscaling*. La sección 3 muestra varias pruebas con el fin de validar la propuesta. Finalmente, las conclusiones obtenidas de este estudio se muestran en la sección 4.

El capítulo 5 presenta el análisis e implementación de las funciones *serveless* dirigidas por eventos para codificar segmentos de vídeo y calcular, opcionalmente, la calidad de los vídeos codificados. En la sección 1 se incluye una introducción al uso de funciones *serverless* para la ejecución de tareas en entornos de vídeo *streaming*. La sección 2 se presentan nuestras funciones de codificación de vídeo sin servidor basadas en FaaS (Function as a Service). La sección 3 analiza la escalabilidad y el consumo de recursos de las funciones propuestas en un banco de pruebas real con vídeos 4K. Para ello, se realiza una comparación del comportamiento entre réplicas *fat* (con seis vCPU) y *slim* (con una sola vCPU). Se evalúa el rendimiento de la propuesta para la codificación multiresolución. Además, se presenta y analiza un *pipeline* para codificar segmentos y calcular la calidad del segmento codificado en base a los eventos de la Nube. La sección 4 presenta una comparación cualitativa de nuestra propuesta con trabajos anteriores sobre codificación de vídeo con *severless*. Por último, en la sección 5 se presentan las conclusiones extraídas de este capítulo.

El diseño y desarrollo de la herramienta *open source* que integra varias métricas de calidad de vídeo, la describimos en el capítulo 6. La sección 1 muestra la importancia de este tipo de herramientas en el desarrollo y funcionamiento de los sistemas de *streaming* actuales. En la sección 2 describe la integración de las métricas VQA en una imagen de contenedor. La sección 3 analiza el consumo de recursos (CPU y memoria RAM) de la herramienta por VQA, y para finalizar en la sección 4 se muestran las conclusiones.

Finalmente, en el capítulo 7 mostramos un compendio de las conclusiones vistas en cada capítulo, ya que cada uno de ellos tiene sus propias conclusiones. Además, en este capítulo describimos algunos trabajos futuros relacionados que podrían ser llevados a cabo con nuestro método de codificación, arquitectura *serverless* y herramienta para calcular la calidad de vídeo.

Capítulo 2

Codificación multimedia en la Nube: revisión y aproximaciones

Si he logrado ver más lejos [que otros], es porque me he subido a hombros de gigantes.

Sir Isaac Newton

Resumen:

Este capítulo revisa los trabajos de investigación relacionados con técnicas de codificación y transcodificación en la Nube, prestando especial atención a la evolución del *streaming* y la importancia del proceso de codificación. Además, se describe la infraestructura en la Nube para escenarios multimedia, y se clasifican 49 artículos válidos. Asimismo, se discuten los estudios relacionados con las aproximaciones en la preparación de contenido de vídeo para la transmisión de VOD, utilizando el estándar DASH, abordadas en esta tesis.

2.1. Introducción

El sector multimedia está experimentando un rápido crecimiento a medida que más empresas y organizaciones recurren a vídeos y/u otros contenidos multimedia para comunicarse con sus clientes, empleados y otras partes interesadas. Para satisfacer la creciente demanda de contenidos multimedia, se necesita una plataforma flexible y escalable que pueda gestionar las grandes cantidades de datos y tráfico asociadas a la transmisión de vídeo y otros contenidos multimedia. La Nube ofrece una solución ideal para ello, ya que ofrece una serie de características que se adaptan bien a las necesidades del sector multimedia.

Algunas de las principales ventajas de la Nube para el sector multimedia son:

- Escalabilidad: Las aplicaciones sustentadas en la tecnología de Nube disponen de la facultad de escalar tanto vertical como horizontalmente con facilidad para ajustarse a fluctuaciones en la demanda. Esto proporciona a las empresas la capacidad para manejar eficazmente los incrementos imprevistos en el tráfico, optimizando así el rendimiento y la disponibilidad del sistema.
- Gestión de recursos: Las infraestructuras basadas en la Nube ofrecen una amplia gama de servicios gestionados que van desde bases de datos, mensajería, pasarelas API, DNS, almacenamiento, etc.
- Reducción de costos: Para empresas emergentes, optar por aplicaciones basadas en la nube resulta estratégico. Este modelo ahorra costos iniciales en hardware e infraestructura, permitiendo un pago solo por los recursos consumidos, fomentando así eficiencia y escalabilidad.

En general, el potencial de las arquitecturas *cloud* las convierte en una opción atractiva para las empresas del sector multimedia que buscan ofrecer contenidos de alta calidad a sus clientes y usuarios.

Para conseguir una reproducción de *streaming* fluida e ininterrumpida, es importante que la tecnología utilizada esté diseñada para transmitir un alto flujo de datos y preparada para soportar variaciones en la demanda del contenido. Una infraestructura elástica basada en la Computación en la Nube (en inglés, *Cloud Computing*) y su evolución hacia la moderna Computación

en la Frontera de la Red Móvil (*Mobile Edge Computing* - MEC); donde la Nube se extiende hasta el borde (en inglés, Edge) de la red, se caracteriza por proporcionar recursos a la medida de cada necesidad, sin un aprovisionamiento excesivo o inadecuado, de forma automática. Por otro lado, al procesar el vídeo, por ejemplo haciendo diferentes representaciones del mismo para diferentes dispositivos o condiciones de red, tenemos más control sobre la calidad de emisión desde el lado del cliente (Zhu et al., 2011).

La agilidad y la elasticidad de la Nube y del *Edge* permiten ampliar la capacidad en función del número de usuarios que tengamos conectados o incluso escalar recursos frente a la rigidez de infraestructuras no basadas en arquitecturas Cloud y MEC.

Además, la relación entre coste y recursos computacionales que ofrecen las infraestructuras Cloud y MEC, junto con la posibilidad de pagar sólo por los recursos utilizados, evitando problemas de sobredimensionamiento, hacen de estas soluciones la mejor opción a la hora de soportar todas las tareas que requieren los proyectos de entrega de contenidos basados en *streaming* (He et al., 2014).

El resto del capítulo se organiza de la siguiente manera. La sección 2.2 presenta una breve visión de la codificación de vídeo en la Nube y las arquitecturas *Edge*. La sección 2.3, se detalla la técnica empleada en la revisión sistemática de la literatura para seleccionar los artículos que se clasifican en la sección 2.4. En esta sección, se presenta una clasificación de los artículos seleccionados según el tipo de tecnología utilizada y su finalidad, y se valida su relevancia mediante el número de referencias. Asimismo, en la sección 2.5 se presentan los trabajos relacionados referentes a las aproximaciones implementadas en esta tesis doctoral. Por último, en la sección 2.6, se presentan las conclusiones del capítulo.

2.2. Estado del Arte

Esta sección ofrece inicialmente una visión general de la codificación de vídeo, la computación en la nube y las arquitecturas *Edge*, así como las principales motivaciones para converger estos dos paradigmas. La sección se estructura en tres partes. En la primera, exploramos la evolución del *streaming*, centrándonos en los métodos más recientes y el papel crucial de la codificación en los sistemas de *streaming* actuales. En la segunda, discuti-

remos la relevancia de los sistemas en la nube en contextos multimedia y describiremos detalladamente la arquitectura e infraestructura que se utiliza en la actualidad para proveer servicios de *streaming* a través de la nube. Finalmente, en la última parte, cotejaremos nuestra investigación con estudios anteriores para evidenciar el progreso que representa.

2.2.1. Streaming Multimedia: Concentrándose en el proceso de codificación

Los sistemas de *streaming* han evolucionado a lo largo de los años a medida que la tecnología y las conexiones de red han mejorado. Desde los primeros sistemas de *streaming* en los años 90 hasta los sistemas de *streaming* actuales, como YouTube o Twitch, todas las tecnologías asociadas a estos sistemas han evolucionado significativamente. Dos claros ejemplos que se tratarán en esta subsección son: a) los protocolos de *streaming* y b) los sistemas de codificación de vídeo.

En el caso de los protocolos de *streaming*, los principales que merecen ser analizados son:

- *Real-time Transport Protocol* (RTP)/*RTP Control Protocol* (RTCP): RTP/RTCP se describen en el RFC 3550 (Schulzrinne et al., 2003). Ambos funcionan sobre conexiones UDP (User Datagram Protocol). Esto significa que el protocolo debe soportar las pérdidas de paquetes, ya que no se garantiza que todos los paquetes llegarán a su destino. Mientras que RTP transporta flujos multimedia (p. ej., audio y vídeo), RTCP se utiliza para monitorizar estadísticas de transmisión, calidad de servicio y facilitar la sincronización de múltiples flujos. A lo largo de los años, ambos protocolos se han considerado el principal estándar para la transmisión multimedia en tiempo real en redes IP.

- *Real Time Streaming Protocol* (RTSP): La última versión de RTSP se define en el RFC 7826 (Schulzrinne et al., 2016). RTSP es un protocolo a nivel de aplicación, similar a HTTP en términos de funcionamiento y sintaxis, pero su objetivo es la transferencia de datos de control multimedia en tiempo real. Se utiliza para establecer y controlar las sesiones multimedia entre los puntos finales, RTSP es un protocolo de control de canales entre el cliente y el servidor multimedia, mientras que el flujo multimedia sigue siendo enviado por RTP.

- *Real-Time Messaging Protocol* (RTMP): RTMP es un protocolo a nivel de aplicación diseñado para multiplexar y empaquetar flujos de transporte multimedia (como audio, vídeo y contenido interactivo) a través de un protocolo de transporte adecuado (por ejemplo, TCP), véase (Lei et al., 2012). Con el objetivo de entregar los flujos de manera fluida y transmitir la mayor cantidad de información posible, RTMP divide los flujos en fragmentos (64 bytes para el audio y 128 bytes para el vídeo por defecto), y el tamaño de estos fragmentos se negocia dinámicamente entre el cliente y el servidor.

Tras examinar algunos de los protocolos de *streaming* más utilizados, es importante considerar ciertos inconvenientes. En la actualidad, la mayoría de los contenidos en Internet se entregan a los clientes mediante redes de distribución de contenidos, y gran parte de ellas no admiten flujos RTP/RTCP. Las principales razones para no soportar los protocolos mencionados son: a) diversos cortafuegos de Internet no permiten este tipo de paquetes, b) en ocasiones, el flujo multimedia necesita múltiples protocolos para funcionar correctamente, como RTSP + RTP/RTCP, y c) los servidores deben administrar una sesión para cada usuario final. A pesar de esto, algunos trabajos han seguido utilizando la capacidad de RTP para la entrega de vídeo, especialmente en el contexto de la Comunicación en Tiempo Real en la Web (WebRTC) (DASH-IF, 2022; Blum et al., 2021; Petrangeli et al., 2019). Este marco tecnológico surgió en 2011 como otra alternativa para las transmisiones en tiempo real. WebRTC es un conjunto de estándares del W3C y el IETF que posibilita la transmisión de contenidos en tiempo real entre usuarios, logrando una latencia de extremo a extremo inferior a medio segundo.

Otra alternativa, a los protocolos anteriormente descritos, es el *streaming* a través de HTTP, que ha evolucionado a lo largo de los años. Al principio, se empleaba una técnica denominada descarga progresiva, que permitía la descarga y reproducción simultáneas del vídeo. Los principales inconvenientes de este tipo de *streaming* radicaban en que el usuario no tenía la opción de iniciar la reproducción del vídeo en un momento específico, y estaba limitado a visualizar únicamente la parte previamente descargada. Además, esta técnica ofrecía la misma calidad de vídeo a clientes con diferentes capacidades de cómputo y conexiones de red, lo que provocaba interrupciones indeseadas en la reproducción.

La evolución de este sistema fue el *HTTP Pseudo-streaming*, que intenta simular el *streaming* bajo demanda al agregar la capacidad de reproducir desde un punto específico del vídeo. En este caso, las partes que se omiten

no se descargan, lo que optimiza el uso del ancho de banda. Sin embargo, el inconveniente es que requiere implementaciones específicas tanto en el cliente como en el servidor.

Hoy en día, el *HTTP Adaptive Streaming* (HAS) es la tecnología de vídeo predominante para la mayoría de las plataformas de *streaming*. En HAS, el contenido de vídeo se divide en pequeños fragmentos llamados segmentos, S_j, que generalmente tienen una duración de uno a diez segundos (Bitmovin, 2021), ya sea con segmentación fija o variable (Schwarzmann et al., 2020a). Se crean representaciones del vídeo con varios niveles de calidad, Q_j. Cada uno de estos segmentos se codifica o transcodifica utilizando diferentes resoluciones, tasas de bits, códecs, etc. La información sobre el contenido generado, como los perfiles de vídeo, metadatos, códecs, rangos de bytes, direcciones IP del servidor y URL de descarga, se describe en un archivo de manifiesto (*Media Presentation Description* -MPD-). Tanto el MPD como los segmentos se almacenan en un servidor HTTP estándar.

En una sesión de *streaming*, un cliente HAS inicia la sesión descargando el archivo MPD desde el servidor HTTP y, luego, en función de las condiciones de la red, comienza a solicitar segmentos de vídeo lo más rápido posible para llenar el búfer[1] (normalmente utilizando la tasa de bits más baja al principio). Una vez que el búfer está lleno, el reproductor continúa descargando nuevos segmentos periódicamente, según el algoritmo *Adaptive Bitrate Streaming* (ABR) seleccionado. Uno de los objetivos de estos algoritmos ABR es evitar que el búfer de reproducción del cliente se vacíe, mientras se maximiza la calidad de experiencia percibida por el usuario. En el trabajo desarrollado por Kua et al. (2017), se presenta una clasificación de algunas técnicas de adaptación del lado del cliente, como *Throughput-Based ABR*, *Buffer-Based ABR* e *Hybrid/Control Theory-Based ABR*.

La principal ventaja de HAS sobre la descarga progresiva y el *streaming* en tiempo real tradicional radica en su capacidad para adaptar la calidad del vídeo según el ancho de banda disponible, con el objetivo de evitar congelamientos (también conocidos como eventos de stalling) en la reproducción. En consecuencia, HAS permite la transmisión de vídeo a través de una red, optimizando el uso del ancho de banda disponible. Además, los flujos de vídeo basados en HTTP pueden atravesar fácilmente cortafuegos y reutilizar la infraestructura HTTP previamente desplegada, como servidores HTTP,

[1] En el *streaming* de audio o vídeo por Internet, se utiliza un búfer para reducir la probabilidad de interrupciones en la reproducción cuando el ancho de banda disminuye o se corta (Wikipedia, Búfer).

proxies HTTP y nodos en la red de distribución de contenidos (Huysegems et al., 2015), tal como se muestra en la Figura 2.1.

Existen diversas especificaciones de formato HAS, siendo las más utilizadas HTTP Live Streaming (HLS) y Dynamic Adaptive Streaming over HTTP (DASH) (Thang et al., 2012). La implementación de DASH se ha comparado con sistemas propietarios populares como Adobe HTTP Dynamic Streaming (HDS), Microsoft Smooth Streaming (MSS) y HLS, demostrando ser un estándar sólido y maduro, especialmente en entornos vehiculares (Müller et al., 2012). Además, DASH es compatible con una amplia variedad de códecs de vídeo y audio, lo que permite una mayor compatibilidad con dispositivos y navegadores. Para ofrecer contenidos con la mejor calidad de experiencia posible, estos sistemas de *streaming* requieren de enfoques de codificación de vídeo eficientes (Seufert et al., 2015).

La codificación de vídeo ha experimentado mejoras significativas a lo largo de los años; sin embargo, el aumento en las resoluciones de los dispositivos y/o los nuevos tipos de contenidos exigen códecs más avanzados, que ofrezcan una mayor capacidad de compresión sin afectar la calidad del vídeo de manera apreciable. Al desarrollar un códec, no solo es crucial la calidad final, sino también su rendimiento, consumo de recursos, tiempo de codificación y tamaño final del archivo.

H.264 (Kalva, 2006) es el códec más utilizado en diversos tipos de transmisiones: según el informe de Bitmovin (2021), el 83 % de los encuestados lo emplean en producción. Creado en 2003, tenía como objetivo reemplazar al tradicional MPEG-4, siendo denominado MPEG-4 AVC. Este formato se utiliza actualmente en películas Blue-Ray, servicios de *streaming*, transmisión por satélite e incluso en la televisión digital terrestre. No obstante, con el surgimiento de resoluciones de vídeo de 4K y 8K, se detectó que H.264 demandaba un gran ancho de banda. Por ello, surgió la necesidad de desarrollar nuevos códecs capaces de reducir la tasa de bits en este tipo de vídeo sin comprometer significativamente la calidad. Como resultado, emergieron H.265 y VP9.

El códec H.265 (Li et al., 2014), también conocido como High Efficiency Video Coding (HEVC), fue desarrollado conjuntamente por el Video Coding Experts Group (VCEG) y el Moving Picture Experts Group (MPEG) en 2013. Se trata de un estándar con derechos de autor, por lo que requiere que los fabricantes de hardware paguen una cuota de licencia para añadir el soporte, y que los desarrolladores paguen una cuota para implementarlo. H.265 permite reducir el ancho de banda necesario para la transmisión de

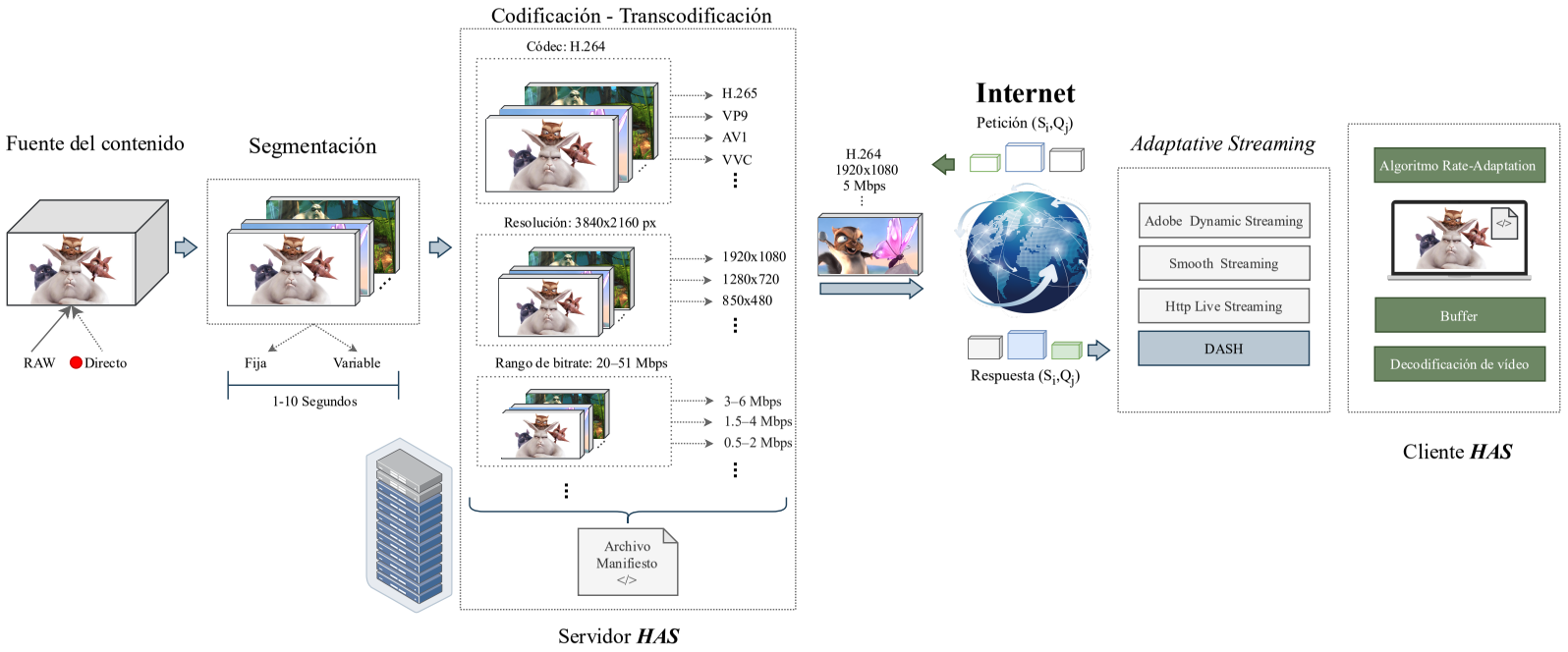


Figura 2.1: HTTP Adaptative Streaming (HAS)

vídeo a casi la mitad en comparación con H.264; manteniendo la calidad. Hay que tener en cuenta también que H.265 puede soportar resoluciones de hasta 8K UHD (8192 × 4320), y una frecuencia de imagen de hasta 300 fps.

VP9 (Mukherjee et al., 2013) es la alternativa de código abierto de H.265 y está libre de derechos, desarrollada por Google como sucesora de VP8. Surgió en 2013 y ha tenido una gran repercusión debido a la gran implantación de Google en su servicio YouTube y en el sistema operativo Android. Los objetivos de diseño de VP9 incluían la reducción de la tasa de bits en un 50 % en comparación con VP8, manteniendo la misma calidad de vídeo, y buscando una mayor eficiencia de compresión que el estándar MPEG HEVC. VP9 está adaptado para resoluciones de vídeo superiores a 1080p (1920×1080 píxeles) hasta 8K con una tasa máxima de 120 fps.

Para comprimir un vídeo, generalmente se utiliza una técnica llamada *motion-compensated prediction*, mediante la cual se codifican bloques de píxeles (cuadrados y/o rectangulares) con referencia a otra zona en el mismo fotograma (intra-predicción), o en otro fotograma (inter-predicción). Con HEVC, se pueden definir bloques grandes, de hasta 64 × 64 píxeles, cambiando el tamaño según la textura. VP9 funciona de forma similar, permitiendo crear macrobloques de hasta 64 × 64 píxeles, pero también permite definir bloques rectangulares de 64 × 32 o 4 × 8 para ganar eficiencia según la necesidad. Cuando el vídeo codificado se descomprime y se reproduce, entran en juego una serie de algoritmos de predicción que permiten reconstruir los píxeles de cada bloque a su estado original. H.264 ejecuta 9 algoritmos de predicción mientras que H.265 ejecuta 35 predicciones para devolver los píxeles a la forma más cercana posible al original. En el caso de VP9, se ejecutan 10 algoritmos de predicción para reconstruir el vídeo.

Con el fin de mejorar VP9, nació AV1. La versión estable del codificador AV1 se publicó en marzo de 2018 (Han et al., 2021). Se trata de un formato de compresión de vídeo abierto y libre de derechos desarrollado por la Alliance for Open Media (AOMedia), un consorcio de las empresas más importantes relacionadas con Internet y el streaming. Este codificador tiene una capacidad de compresión de más del 30 % en comparación con sus predecesores (VP9 y H.265) (Chen et al., 2018).

En AV1, el contenido de los fotogramas se divide en bloques adyacentes del mismo tamaño denominados superbloques (similares a los macrobloques). Los superbloques tienen forma cuadrada y pueden tener un tamaño de 128 × 128 o 64 × 64 píxeles, pero estos superbloques pueden dividirse en bloques más pequeños hasta llegar a los 4 × 4. AV1 introduce el concepto de *T-shaped*

que permite dividir un superbloque en divisiones horizontales o verticales en cuatro franjas de relación de aspecto 4:1 y 1:4. Amplía los modos intra direccionales añadiendo 3 nuevos predictores suaves. Además, AV1 amplía el número de referencias para cada fotograma de 3 en VP9 a 7.

H.266/VVC: Versatile Video Coding (VVC/H.266) (Bross et al., 2021b) es el último estándar de codificación que ofrece un 50 % más de compresión que H.265/HEVC. Aumenta el tamaño de los bloques en el modelo temporal a 128×128, con bloques de partición binaria o ternaria en comparación con la partición cuaternaria de HEVC. Permite una partición diferente para los planos de luminancia y crominancia y permite la aceleración por hardware mediante el procesamiento en paralelo. Para la predicción intrafotograma, se utilizan 67 modos de predicción en lugar de los 33 utilizados en HEVC, además de permitir el uso de bloques rectangulares. La predicción entre fotogramas permite la predicción a partir de dos imágenes de referencia, además de aumentar los grados de libertad de los vectores de movimiento de 2 a 3 dimensiones. Para la transformación se utilizan bloques no rectangulares, realizando diferentes tipos de transformaciones en función del modo de predicción. El valor máximo del QP o cuantificador se incrementa de 51 a 63 para permitir menores tasas de bits.

Hasta ahora, la evolución de los códecs ha sido siempre similar. Se mejora la calidad manteniendo el bitrate a costa de tener codificadores que requieren mayor capacidad de cálculo. Pero, ¿qué ocurre si estos codificadores de nueva generación quieren funcionar en dispositivos *Edge* o IoT? Para ello, MPEG/ISO ha desarrollado el estándar MPEG-5 Parte 2 LCEVC (Low Complexity Enhancement Video Coding) (Battista et al., 2022). MPEG-5 Parte 2 LCEVC es la primera norma de mejora acreditada internacionalmente para cualquier esquema de compresión de vídeo existente y futuro. Mejora el rendimiento de la compresión de cualquier códec de vídeo básico (por ejemplo, AVC, HEVC, AV1, EVC o VVC) y ofrece una calidad de imagen mejorada con una tasa de bits hasta un 40 % menor, tanto para la transmisión en directo como para el VOD.

La Figura 2.2 muestra el estado actual de los códecs más importantes H.264, H.265, VP9, AV1 y VVC realizando una pequeña comparación entre ellos basada en las siguientes cuatro características (ver (Grois et al., 2016) y (García-Lucas et al., 2020)):

- Rendimiento en la codificación: Tiempo de codificación de un vídeo con parámetros de codificación similares.

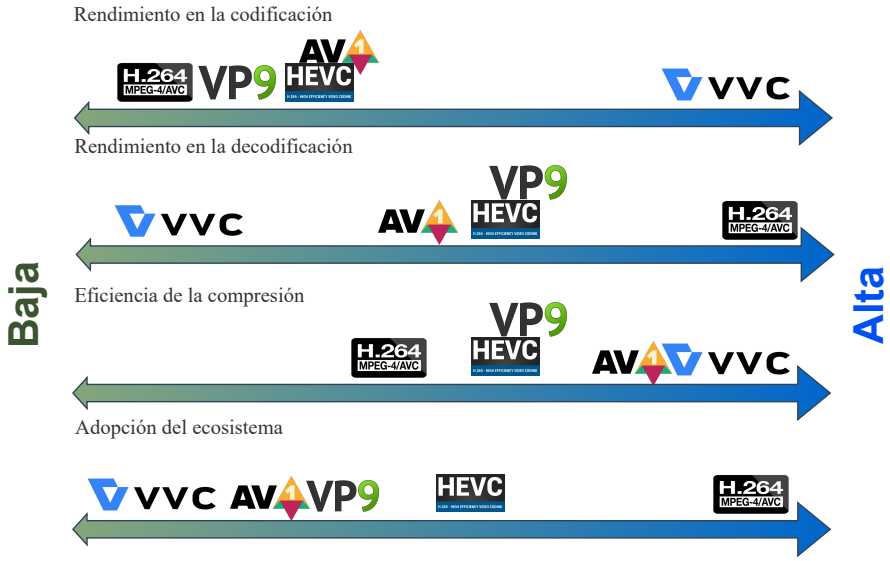


Figura 2.2: Situación actual de los códecs: una pequeña comparación.

- Rendimiento en la decodificación: Velocidad de un flujo de bits determinado cuando se reproduce en un dispositivo final.
- Eficiencia de la compresión: Medidas de calidad subjetivas y objetivas para una tasa de bits determinada.
- Adopción del ecosistema: Grado de implementación y actualización de codificadores y decodificadores.

2.2.2. Arquitecturas de la Nube Multimedia y de la Computación en la Frontera de la Red Móvil

Los servicios *Over-The-Top* actuales requieren una infraestructura ágil para escalar instantáneamente y hacer frente a las grandes fluctuaciones que demandan estos tipos de servicios. La computación en la Nube y en el *Edge* ayudan a los servicios de *streaming* a resolver estos desafíos. La computación en la Nube ha sido la evolución del uso de la virtualización para ofrecer como servicio una arquitectura multiarrendatario que puede ser utilizada simultáneamente por múltiples clientes (arrendatarios). La computación en el *Edge* ha sido la evolución de la infraestructura de la Nube para cubrir no

sólo el segmento de red del centro de datos sino también la última milla cerca de los clientes finales.

La Nube Multimedia y la computación en el *Edge* tienen muchas similitudes con la computación en Nube de uso general (Zhu et al., 2011). Sin embargo, los servicios multimedia son muy diversos. Existen numerosos tipos de medios y servicios asociados, como la voz sobre IP (VoIP), la transmisión de vídeo, la compartición y edición de fotos y vídeos, y la búsqueda de imágenes, por nombrar algunos; todos estos tipos de medios y servicios deben soportar un gran número de usuarios al mismo tiempo. Además de la heterogeneidad de los servicios, los distintos tipos de dispositivos, como ordenadores, televisores inteligentes y teléfonos inteligentes, tienen distintas capacidades de procesamiento multimedia. La Nube/*Edge* debe ser capaz de adaptarse a los distintos dispositivos en términos de CPU y GPU, pantalla, memoria, almacenamiento y ancho de banda. Para satisfacer sus necesidades específicas de QoS, la infraestructura también debe soportar el aprovisionamiento de QoS. Hay dos enfoques para habilitar el aprovisionamiento de QoS para multimedia. Por un lado, añadir la QoS a la infraestructura existente de la Nube y al *Egde*. Por otro lado, añadir una capa de *middleware*[2] de QoS entre la infraestructura de la Nube y las aplicaciones multimedia. Por ejemplo, (Al-hammouri et al., 2018) propone un *middleware* de capa de aplicación como una solución para aumentar la fiabilidad y la flexibilidad de la red, en las aplicaciones multimedia en tiempo real que utilizan el método de codificación de vídeo conocido como Scalable Video Coding (H264/SVC) o codificación por capas.

En definitiva, la gran demanda de datos multimedia en Internet podría congestionar una arquitectura de Nube/Edge de propósito general si no se la ha dotado adecuadamente de soporte para las capacidades multimedia. Las arquitecturas actuales de Nube y Edge se han concentrado principalmente en proporcionar recursos de computación y almacenamiento. En la actualidad, estas arquitecturas se están perfeccionando con técnicas y mejoras para cumplir los requisitos de calidad de servicio, como el aumento del ancho de banda, la minimización de la latencia y el jitter, etc., y con nuevos mecanismos para cumplir los requisitos de calidad de servicio, como el *Network Slicing* (corte de red)[3] (Escolar et al., 2021).

[2]Middleware: Software que se sitúa entre el sistema operativo y las aplicaciones que se ejecutan en él. Permite la comunicación y la administración de datos para aplicaciones distribuidas, como las aplicaciones basadas en la Nube (...). Ejemplos de middleware son los servidores web, los servidores de aplicaciones y los sistemas de administración de contenido (Azure, 2022).

[3]Corte de red: Es una red lógica independiente de extremo a extremo que funciona

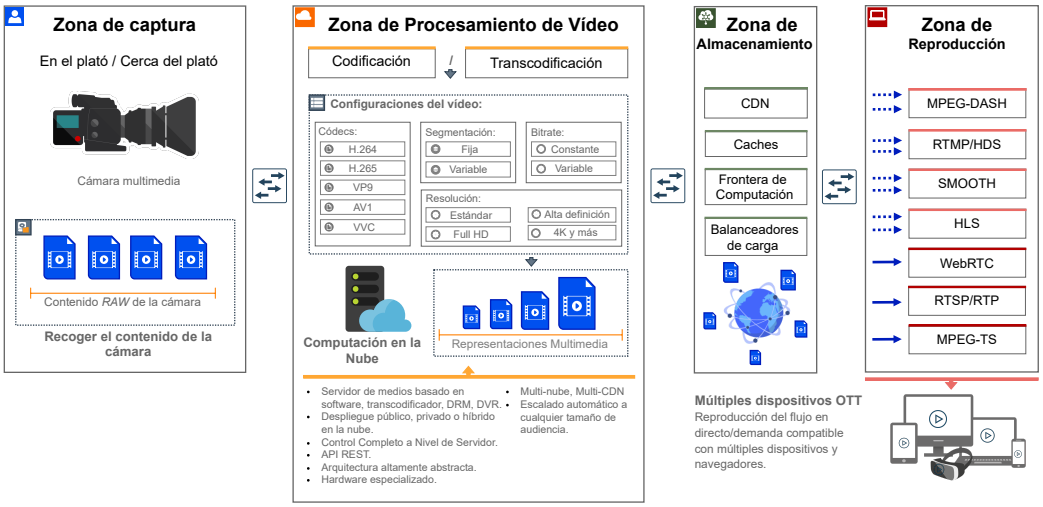


Figura 2.3: Arquitectura de la Nube multimedia y la Frontera de Computación.

Independientemente de la infraestructura informática utilizada, los procesos de *streaming* pueden dividirse en 4 pasos (véase la Figura 2.3):

En la **Zona de Captura**, se pueden llevar a cabo dos tareas, dependiendo del tipo de *streaming* (VOD o en directo). En el caso del VOD, aquí es donde se realizan las tareas de producción y postproducción del contenido. Estas tareas se efectúan con vídeos en formato *Raw* (sin comprimir) o de muy alta calidad para preservar la calidad de la imagen. En el caso de los sistemas en directo, en esta zona se lleva a cabo la ingesta de contenidos multimedia provenientes de unidades móviles de emisión, así como la producción en tiempo real. En ambos escenarios, el flujo multimedia de muy alta calidad se envía al siguiente paso, la Zona de Procesamiento de Vídeo. Un vistazo en profundidad sobre cómo Netflix ofrece tecnología e infraestructura basada en la Nube para ayudar a los equipos de producción a crear e intercambiar contenido durante las etapas de producción y postproducción se puede encontrar en (Anton et al., 2021).

En la **Zona de Procesado de Vídeo**, se realizan las tareas de codificación y transcodificación según el tipo de stream procedente de la Zona de Captura. Las codificaciones se pueden realizar para diferentes codecs (AVC, HEVC, VP9, AV1, etc.), para diferentes tasas de bits, diferentes resolucio-

en una infraestructura física compartida, capaz de proporcionar una calidad de servicio negociada (...) transparente para los clientes (GSM, 2017).

nes, etc. Todo ello pensado para el tipo de protocolo de *streaming* que se vaya a utilizar (por ejemplo, DASH o HLS). Estos procesos deben realizarse en paralelo para reducir el retardo y poder ofrecer todas las versiones del contenido al usuario final y mejorar así la calidad percibida.

La **Zona de Almacenamiento** es clave para ofrecer un buen servicio al usuario final. Los sistemas de Nube actuales cuentan con sistemas de almacenamiento distribuido que acercan el contenido a los usuarios. Para evitar sobrecargar los servidores de contenido primario, las arquitecturas CDN también se pueden utilizar para proporcionar contenido multimedia con una latencia reducida. En particular, pueden aprovechar las ventajas de las infraestructuras del *Edge*, que permiten la geolocalización dinámica y proactiva del contenido en la última milla respecto al usuario, lo que reduce significativamente la latencia requerida para la entrega.

Finalmente, la **Zona de Reproducción** debe estar adaptada y ser compatible con los protocolos de *streaming* (HTTP —DASH, HLS—, RTSP, RTP, etc.). Dependiendo del tipo de transmisión, protocolo o dispositivo, los usuarios en esta zona deben utilizar una aplicación específica para acceder al contenido de vídeo. Como se mencionó en la subsección 2.2.1, por ejemplo, en una transmisión que emplea HAS, la aplicación del cliente controla el algoritmo ABR solicitando tasas de bits de vídeo basadas en las condiciones de la red observadas, regulando así la tasa de transmisión entre el cliente y el servidor (Kua et al., 2017).

2.2.3. Trabajos relacionados

En esta subsección vamos a presentar otras revisiones bibliográficas relacionados con el *Multimedia Cloud Computing* (MCC) publicados anteriormente, prestando especial atención a las diferencias entre estos trabajos anteriores y el estudio realizado en gran parte de este capítulo de la tesis doctoral.

El documento (Zhu et al., 2011) presenta los principales conceptos de la computación multimedia en la Nube. Los autores abordan la computación multimedia en la Nube desde dos perspectivas: *multimedia-aware cloud* y *cloud-aware multimedia*. Demostraron cómo la *multimedia-aware cloud* puede proporcionar soporte de QoS, procesamiento paralelo distribuido, almacenamiento y equilibrio de carga para diversas aplicaciones y servicios multimedia. Además, exploraron cómo los recursos de la computación en Nube pueden ser utilizados de la mejor manera por los servicios y aplicaciones multimedia,

como el almacenamiento y la compartición, la producción y el montaje, la adaptación y la entrega, y la renderización y la recuperación.

En (Xu y Mao, 2013), los autores describen cómo utilizar la computación móvil en la Nube para habilitar aplicaciones multimedia en dispositivos móviles. Se centran en el aspecto técnico y discuten los retos y soluciones existentes desde diferentes perspectivas. En este trabajo, los autores prestan atención al consumo de energía de los clientes y a cómo la Nube podría ayudar a reducirlo. Los posibles nuevos retos deben tener en cuenta la calidad de la experiencia, la seguridad y la privacidad. (Wen et al., 2014) es otro trabajo que estudia el paradigma de la Nube multimedia móvil. Este trabajo estudia esta área de investigación, que abarca desde la gestión y el control de los recursos, los servicios de la plataforma de multimedia, los sistemas en la Nube y las aplicaciones. Es el único estudio en el que los autores analizan el proceso de codificación/transcodificación en una subsección denominada *"Media Representation"*. Cabe destacar que el análisis realizado por los autores se fundamenta exclusivamente en siete trabajos previos.

Otra visión general relacionada con la computación multimedia en la Nube es (Huang et al., 2011). En este trabajo, los autores muestran una visión general del sistema de almacenamiento en la Nube y su problema de seguridad. Describen varias ideas y soluciones clave sobre la integridad de los datos, la confidencialidad de los datos, el control de acceso y la manipulación de los datos. Los autores concluyen diciendo que la seguridad del almacenamiento multimedia en la Nube está todavía en sus inicios y esperan una importante mejora en un futuro próximo. Por otra parte, los autores de (Yang et al., 2017) introducen el concepto de protección multimedia con un método basado en el control de acceso por roles. Describen el proceso completo de registro, asignación de roles, la solicitud del de los archivos multimedia, la solicitud de encriptación de datos por parte del propietario, y el inicio de sesión del usuario y el acceso al archivo multimedia, lo que garantiza la seguridad de los archivos multimedia.

El objetivo principal del estudio de (Tselios y Tsolis, 2016) es presentar diversas plataformas, paquetes de software, herramientas de entrega de aplicaciones y arquitecturas líderes que podrían facilitar el despliegue, mantenimiento y escalabilidad de aplicaciones multimedia en entornos de computación en la Nube. Un caso específico, basado en Amazon Web Services (AWS), es explorado en (Toshniwal et al., 2020). Este trabajo examina cómo las tareas de codificación y pre-transcodificación, que requieren una alta capacidad computacional, se trasladan a la Nube para optimizar costes. No

obstante, es importante señalar que estas tareas de transcodificación representan la mayor demanda computacional en todo el proceso de streaming.

Por último, hay otro trabajo (Abdallah et al., 2018), en el que los autores exploran los últimos avances en la computación de vídeo sensible a la latencia en la Nube, que es esencial para los servicios de vídeo conversacional soportados en la Nube, como los juegos en la Nube, la Realidad Virtual (RV), la Realidad Aumentada (RA) y la telepresencia. En este artículo, los autores adoptan un enfoque descendente para abarcar la bibliografía: desde las aplicaciones y experiencias, pasando por la arquitectura y la gestión, hasta la optimización dentro y fuera de la Nube. También señalan los principales retos pendientes y esperan estimular más actividades de investigación en esta nueva y apasionante dirección.

En resumen, aunque existen diversos artículos de revisión enfocados en la computación multimedia en la Nube, ninguno se centra específicamente en los procesos de codificación y transcodificación. Estos procesos, que demandan una alta capacidad computacional, son llevados a la Nube para optimizar costes. El presente estudio aborda este tema en particular, ya que no se han encontrado trabajos similares y se trata de un área de investigación actual en constante evolución, que busca mejorar la calidad de la experiencia para los usuarios finales y reducir los costes asociados a estas tareas computacionalmente exigentes.

2.3. Metodología de investigación

Este trabajo utiliza la metodología de revisión sistemática de la literatura (SLR) propuesta por Kitchenham y Charters (2007). El objetivo es definir las preguntas de investigación adecuadas y buscar los estudios pertinentes que se centren en ellas. La revisión se divide en tres fases: planificación, ejecución e informe de resultados.

Estas etapas tienen como objetivo determinar el estado actual del ámbito de investigación, evaluar las contribuciones y lagunas existentes, y obtener conclusiones parciales sobre cada pregunta de investigación. Con esto, se elabora una conclusión general del informe. En las subsecciones siguientes, se describirán con mayor detalle cada una de estas etapas.

A su vez, se han tenido en cuenta las directrices de Petersen et al. (2008a),

en particular a la hora de clasificar un estudio e identificar su área de investigación. Los pasos de la investigación utilizados por Kitchenham y Charters (2007) y Banijamali et al. (2020) se muestran en la Figura 2.4. Por ejemplo, en esta investigación se llevó a cabo un estudio piloto adicional para examinar las posibles cadenas de búsqueda (similares a Kitchenham y Charters (2007)).

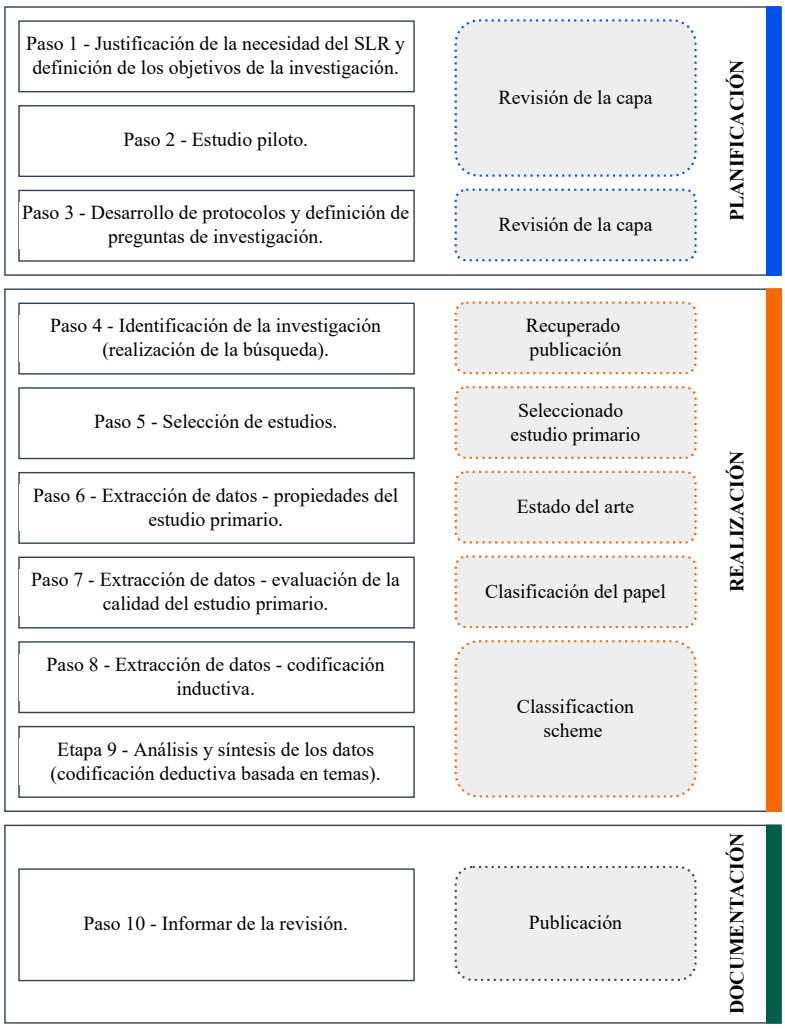


Figura 2.4: Pasos de la revisión sistemática de la literatura.

2.3.1. Identificación de la necesidad de la revisión

Ninguna de las revisiones analizadas en la subsección 2.2.3 se centra en una revisión sistemática de la literatura enfocados de las propuestas de codificación de vídeo sobre infraestructuras en la Nube. Para realizar esta revisión se ha llevado a cabo un proceso sistemático de búsqueda y evaluación de fuentes primarias sobre el tema tratado. En este informe se aportan datos estadísticos de los trabajos seleccionados y una clasificación de los mismos que puede ser utilizada posteriormente en otros proyectos de investigación para alcanzar un conocimiento sólido sobre el estado actual del problema tratado.

2.3.2. Especificar el objetivo y las preguntas de la investigación

Para identificar de manera sistemática el propósito de investigación de este estudio, utilizamos la técnica *Goal-Question-Metric* (GQM) (Van Solingen y Berghout, 1999). Un objetivo GQM es un objetivo de medida que ha sido formalizado de acuerdo a criterios específicos (Van Latum et al., 1998), el cual se muestra en la Tabla 2.1.

Tabla 2.1: Objetivo de la investigación.

Analizar	el diseño de los sistemas de codificación basados en la Nube
Con el propósito de	caracterizar
respecto al	tipo de arquitectura de computación en la Nube, tipos de servicios en la Nube y tipos de *streaming*
Desde la perspectiva de	el investigador
En el contexto de	la convergencia de la codificación/transcodificación de vídeo en la computación en la Nube y las arquitecturas MEC.

El objetivo general de esta investigación se define como sigue:

Analizar la implementación de los diseños de codificación basados en la Nube **con el propósito** de caracterizarlos en relación con el tipo de computación en Nube, los tipos de servicios en Nube y el tipo de *streaming*

desde la perspectiva del investigador **en el contexto de** la convergencia de la codificación/transcodificación de vídeo en arquitecturas de computación en la Nube y MEC. Como se muestra en la Tabla 2.2, este objetivo se dividió en 4 preguntas de investigación y sus fundamentos.

Tabla 2.2: Preguntas de investigación (RQs)

ID	Pregunta	Racional
RQ1	¿Hay muchos trabajos sobre codificación de vídeo distribuido en entornos en la Nube?	Analizar si hay muchos trabajos sobre el tema y su relevancia, debido a la gran demanda de este tipo de contenido multimedia.
RQ2	¿Qué tipo de arquitectura en la Nube se ha utilizado para implementar cada solución?	Identificar y sintetizar la arquitectura o paradigma utilizado en las propuestas seleccionadas.
RQ3	¿Qué códecs de vídeo se han utilizado en los experimentos presentados en cada artículo?	Identificar los códecs de vídeo utilizados y el número de códecs utilizados por cada artículo.
RQ4	¿Las soluciones están destinadas a vídeo bajo demanda, transmisión en vivo o ambos?	Identificar y estructurar los estudios primarios en términos de tipos de transmisión de vídeo.
RQ5	¿Cómo pueden evolucionar los sistemas de codificación de vídeo a lo largo de los nuevos paradigmas de la computación en la Nube?	Identificar y argumentar los posibles nuevos desafíos relacionados con el tema, y hacia dónde pueden encaminarse las nuevas soluciones de sistemas de codificación de vídeo en los próximos años.

2.3.3. Definición de los criterios de inclusión y exclusión

Seleccionamos estudios para su inclusión en la revisión bibliográfica si presentaban una contribución científica al cuerpo del conocimiento sobre la codificación de vídeo en el contexto de la computación de vídeo en la Nube y la computación en el *Edge*. Específicamente, incluimos artículos científicos en el contexto de la convergencia de la Nube/*Edge* y la preparación de contenidos de vídeo que cumplieran todos los criterios siguientes:

1. implementar un sistema de codificación o transcodificación de vídeo en la Nube,
2. definir un método de codificación o transcodificación de vídeo,
3. si la propuesta ha sido implementada y probada en un entorno de Nube privada, pública o entorno de Nube federada, y
4. si el año de publicación del trabajo es después de 2010.

En consecuencia, se consideraron los siguientes criterios de exclusión:

1. estudios que se ocupan de la preparación de contenido de vídeo, computación en la Nube o computación en la red, pero no de su convergencia;
2. estudios que tratan temas distintos a la codificación o transcodificación de vídeo;
3. artículos duplicados;
4. estudios no revisados por pares, incluyendo introducciones a *special issues*, llamadas para artículos, discursos de apertura, prefacios, estándares, patentes, etc;
5. Se han excluido los trabajos que son una extensión de un trabajo anterior o los trabajos que no están escritos en inglés.

2.3.4. Definición de la estrategia de búsqueda (Meta-Análisis)

En la revisión sistemática de la literatura, la identificación de estudios relevantes que respondan a las preguntas de investigación es un paso importante (Kitchenham y Charters, 2007). Por ello, para desarrollar y analizar la estrategia de búsqueda, los investigadores utilizan diferentes pautas o métodos (Petersen et al., 2008b).

En este trabajo se ha optado por realizar una revisión progresiva de la información, aplicando secuencialmente los métodos de búsqueda automática y bola de nieve según las directrices de (Wohlin, 2014).

Para iniciar el trabajo, se realizó un estudio piloto para definir la estrategia de búsqueda óptima que minimizara el ruido y recuperara adecuadamente

los estudios relevantes. El estudio piloto comenzó definiendo un conjunto de palabras clave con la siguiente cadena de búsqueda: *("cloud" OR "cloud-based", "architecture") AND ("video" OR "media" OR "multimedia") AND ("coding" OR "encoding" OR "transcoding")*. La cadena seleccionada por los autores para abordar los temas relacionados con la investigación se agrupa en tres elementos con cierto grado de afinidad y cada grupo se relaciona entre sí mediante el operador lógico "AND".

Para llevar a cabo una evaluación objetiva de la cadena de búsqueda, se empleó Google Scholar. En este contexto, se adaptó la cadena a la herramienta de búsqueda avanzada, optando por todos los metadatos como posibles coincidencias y estableciendo un periodo de búsqueda predefinido en los objetivos de este estudio (2011-2022).

Por consenso en el grupo de trabajo, se listaron los trabajos por relevancia y se excluyeron los trabajos con patentes. De un total de 443 resultados obtenidos, se seleccionaron los 100 primeros artículos para el estudio piloto. Los artículos fueron analizados por separado por varios investigadores, con el fin de reducir el sesgo a la hora de registrar sus votos con respecto a la relevancia de los artículos.

El estudio piloto dio como resultado 36 artículos relevantes (36 %) en el ámbito de este estudio. Como siguiente paso, se realizó un análisis de los metadatos recogidos de este grupo de artículos. Se notó que la palabra clave más empleada en este campo era "Cloud Computing". La Figura 2.5 muestra los resultados del análisis de las palabras clave según la frecuencia de aparición en el conjunto de estudios relevantes. Permite al lector obtener una visión completa de las palabras clave asociadas al tema.

Además, se puede observar la aparición de determinadas palabras en la mayoría de los títulos de los artículos relevantes, como vídeo (69.4 %), transcodificación (63.9 %) o Nube (41.7 %).

En consecuencia, el estudio piloto realizado nos permitió modificar la cadena de búsqueda sustituyendo la palabra "Arquitectura" por "Cloud Computing". Se mantuvo la misma estructura y orden preestablecido que la cadena de búsqueda base, como puede verse en la Tabla 2.3.

Como paso siguiente, el grupo de trabajo definió un conjunto de bases de datos bibliográficas donde encontrar los artículos relevantes para este estudio. Este conjunto de buscadores contiene un gran número de publicaciones científicas dentro del campo de la informática. Entre las bases de datos se

Keyword	Freq.	Keyword	Freq.	Keyword	Freq.	Keyword	Freq.
Cloud Computing	19	ACO	1	Immersive Media	1	GPU	1
Video Transcoding	9	Hadoop & Mapreduce	1	Content Delivery Networks	1	Video Coding	1
Hadoop	7	Cloud Platform	1	5g Networks	1	Jobs Distribution	1
Mapreduce	6	Multi-modal	1	Stereoscopic	1	PAAS	1
Transcoding	4	Cost-efficiency	1	Panorama	1	Multimedia Transcoding	1
Scheduling	4	Complexity Prediction	1	MV-HEVC	1	HVTS	1
Video Encoding	3	Multimedia	1	Secure Deduplication	1	Multimedia Service	1
Resource Provisioning	3	Locality-aware	1	Scalable Video Coding (SVC)	1	Federation	1
Video Streaming	3	Video Splitter	1	Layer-level Deduplication	1	Distributed System	1
Resource Allocation	3	Cloud Media Center	1	Computation and Storage Tradeoff	1	Segmentation	1
Video Quality	3	Multiple Wireless Interfaces	1	Thermography	1	Admission Control	1
Hyperconvergence	2	Serverless Computing	1	Machine Learning	1	Mobile Streaming	1
Task Scheduling	2	Hadoop and Mapreduce	1	Load Balancing	1	Dynamic Adjustable Encode	1
Scalable Video Coding	2	CSA	1	Quality of Service	1	Profit Maximization	1
Layer-splitting	2	Video Transforming	1	Dynamic Resource Allocation	1	Real Time	1
Multimedia Big Data	2	PSNR	1	Cost Saving	1	eHealth	1
Media Cloud	2	Video Content Delivery	1	Test-zone Search	1	Partial Transcoding Scheme	1
User Viewing Pattern	2	Adaptive Bit Rate	1	Frame Copy	1	Social Media	1
Viewer Behavior	2	Hardware Acceleration	1	Error Concealment	1	Partial Transcoding	1
Encoding	2	Heterogeneous VM Provisioning	1	End-to-End Delay	1	Video Processing	1
Cloud	2	QoS-aware Scheduling	1	Block Matching	1	Video Management	1
Cloud Services	2	On-demand Video Transcoding	1	HD	1	J2EE	1
Streaming	2	Remote Production	1	UHD Video	1	Distributed Video Coding	1
Distributed Processing	1	Prediction	1	Super-resolution	1	Map Reduce	1
Adaptive Interface Selecti	1	Network Functions Virtualization	1	Real-time	1	FFMPEG	1
						Multimodal	1

Figura 2.5: La representación gráfica refleja los resultados derivados del análisis de la frecuencia con la que aparecen las palabras clave dentro del compendio de estudios identificados como relevantes.

Tabla 2.3: Cadenas de búsqueda utilizando el refinamiento de palabras clave.

Cadenas de búsqueda.	
Cadena de búsqueda base:	(“cloud”OR “cloud-based”OR “architecture”) AND (“video” OR “media” OR “multimedia”) AND (“coding” OR “encoding” OR “transcoding”)
Cadena de búsqueda final:	(“cloud” OR “cloud-based” OR “cloud computing”) AND (“video” OR “media” OR “multimedia”) AND (“coding” OR “encoding” OR “transcoding”)

encuentran ACM Digital Library, IEEE Xplore y Scopus. Kitchenham y Brereton (2013) mencionan que la búsqueda en IEEE, ACM garantiza una buena cobertura de revistas y conferencias importantes y al menos dos sistemas de indexación general.

Para realizar la búsqueda automatizada, la cadena de búsqueda se estructuró siguiendo las especificaciones de la herramienta de búsqueda avanzada de cada una de las bases de datos. Sin embargo, se realizó una segmenta-

ción de los metadatos considerados para la recuperación de los artículos. En ACM Digital Library, sólo se utilizó el resumen para la búsqueda ya que consideraba un mayor número de artículos a recuperar que utilizando una combinación de Título + Resumen + Palabras clave. En IEEE Xplore se utilizó el resumen como campo de búsqueda junto con un filtro que limita los temas a considerar, tales como: computación en Nube, codificación de vídeo, streaming de vídeo, compresión de datos, transcodificación, asignación de recursos, aprendizaje (inteligencia artificial) y computación multimedia. Se utilizó Scopus porque incluye plataformas de búsqueda como ScienceDirect e indexa bases de datos externas. Dado que esta plataforma indexa artículos científicos de bases de datos externas, se descartaron los resultados obtenidos de las dos bases de datos anteriores.

2.3.5. Extracción y síntesis de datos

Los resultados obtenidos mediante la búsqueda automatizada utilizando las palabras clave que aparecen en la Tabla 2.3 se procesaron y finalmente se aplicó la búsqueda por bola de nieve para minimizar el riesgo de que se omitieran estudios relevantes.

1. El primer paso en la selección de estudios es la búsqueda automatizada descrita en la subsección anterior. Esta consiste en aplicar las cadenas de búsqueda en las bases de datos de Internet mencionadas para obtener el primer conjunto de estudios preliminares. Al combinar los resultados de todas las fuentes de datos seleccionadas, este paso inicial identificó un total de 867 trabajos. La Biblioteca Digital ACM contribuyó con 80 trabajos, IEEE con 526 y Scopus con 261.

2. El segundo paso consiste en analizar los títulos y eliminar los artículos duplicados. Los trabajos obtenidos en el primer paso se sometieron a los criterios de inclusión/exclusión establecidos previamente en la subsección 2.3.3, considerando también los títulos de cada artículo. Además, se eliminaron los artículos duplicados procedentes de las diferentes bases de datos en línea. Al finalizar este paso, se contabilizaron 664 artículos.

3. El tercer paso es el análisis de los metadatos. Los metadatos tienen en cuenta el resumen y las palabras clave de cada artículo para determinar si cumplen o no los criterios de selección. La aplicación de este paso dio

como resultado una selección de 70 artículos como los más relevantes para este estudio.

4. El análisis del texto completo es el cuarto paso. En ella se examinan los textos completos de los artículos adquiridos a fin de realizar un análisis más profundo de su adecuación a los criterios de selección. A continuación, se escogen los que cumplen los requisitos de inclusión. Al término de este proceso, el conjunto final de artículos constaba de 46 artículos en total.

5. La quinta etapa consiste en aplicar la técnica de búsqueda por bola de nieve, que complementa el primer método de búsqueda. Este enfoque permite identificar aquellos trabajos que no se detectaron durante la búsqueda automática, asegurando así la inclusión de todos los trabajos relevantes. La técnica implica examinar la lista de referencias o citas (bola de nieve hacia atrás) de cada artículo en el conjunto de trabajos y, posteriormente, analizar las citas recibidas por estos artículos (bola de nieve hacia adelante) para descubrir otras fuentes o trabajos primarios.

   El tema tratado en este estudio tiene unos criterios bastante específicos, por lo que no fue posible señalar una colección concreta de artículos para aplicar la técnica de búsqueda por bola de nieve. Sin embargo, debido al número de citas, se eligieron dos estudios como candidatos para emplear esta técnica:

   - "Cloud transcoder: Bridging the format and resolution gap between Internet vídeos and mobile devices" (Li et al., 2012) y
   - "Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing" (Jokhio et al., 2013).

   Como resultado de este paso, se incorporaron 3 nuevos artículos descubiertos mediante la técnica de bola de nieve.

Después de aplicar el método de investigación, se identificaron 49 artículos que cumplían los criterios de selección, publicados entre enero de 2011 y diciembre de 2022. Estos artículos se denominan estudios primarios.

Los pasos realizados en este trabajo, así como los resultados obtenidos en etapa, se presentan en la Tabla 2.4.

Tabla 2.4: Totales de estudios primarios seleccionados de cada paso.

Pasos	Base de Datos	Estudios por Base de Datos	Estudios Seleccionados
1er Paso Búsqueda Automática en Bases de Datos	ACM	80	867[1]
	IEEE	526	
	SCOPUS	261	
2nd Paso Análisis de Títulos y Duplicados	ACM	68	664[2]
	IEEE	416	
	SCOPUS	180	
3er Paso Análisis de Metadatos	ACM	15	70
	IEEE	36	
	SCOPUS	19	
4to Paso Análisis del Texto Completo	ACM	8	46
	IEEE	27	
	SCOPUS	11	
5to Paso	Búsqueda de Bola de Nieve	3	49*

[1] Fecha de búsqueda: 03.12.2022. [2] No se consideraron los documentos duplicados.

2.3.5.1. Control de validez

En consonancia con la técnica de trabajo empleada para definir la cadena de búsqueda, descrita en la subsección 2.3.4, y con el propósito de minimizar el sesgo y la subjetividad en el proceso de selección de artículos, varios investigadores analizaron un conjunto de trabajos y, utilizando su criterio, determinaron qué artículos cumplían o no con los criterios de selección. Finalmente, el autor de esta tesis y sus directores participaron de la segunda a la cuarta etapa del proceso de búsqueda y selección previamente descrito.

Juntos, procesaron los metadatos de los artículos resultantes del primer paso, con el objetivo de identificar los artículos repetidos. Posteriormente, los artículos se distribuyeron aleatoriamente entre todos (33 % para cada uno) para disminuir el sesgo entre las revisiones. Los artículos aceptados por cada uno de ellos en las distintas etapas se incluyeron en el conjunto primario de estudios.

Es importante mencionar que, en la última etapa, todos colaboraron aplicando conjuntamente la técnica de bola de nieve, asegurándose de encontrar y seleccionar correctamente todos los artículos que cumplían con los criterios de inclusión/exclusión.

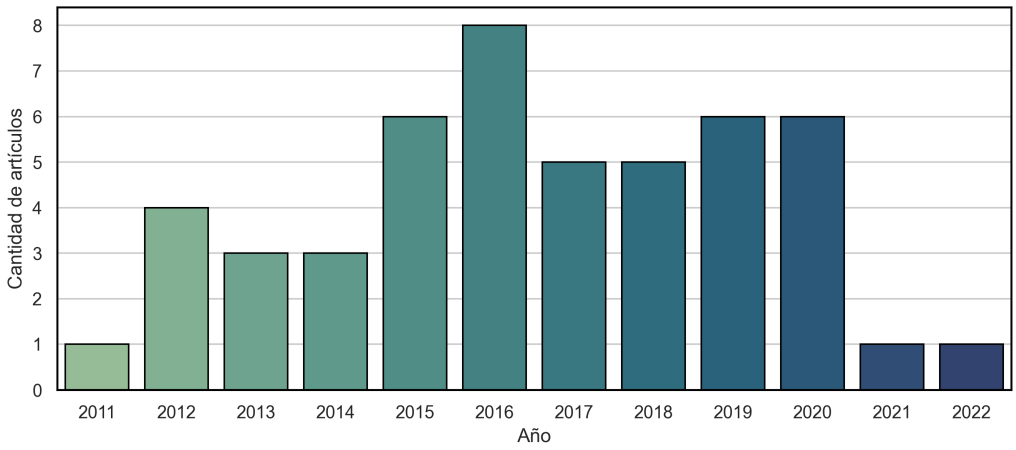


Figura 2.6: Número de artículos relacionados con el tema publicados al año.

2.3.5.2. Informe de resultados

En el apartado 2.4 de este trabajo se presenta un informe de los resultados obtenidos (fase final de la metodología de selección de artículos aplicada) con el objetivo de desarrollar el análisis y las clasificaciones extraídas de los estudios primarios seleccionados.

2.4. Codificación multimedia en la Nube: Revisión

2.4.1. Introducción

Conforme al proceso de búsqueda y selección detallado en la subsección 2.3, se seleccionaron 49 artículos vinculados al proceso de codificación o transcodificación de vídeo en infraestructuras en la Nube. Dichos artículos fueron publicados desde 2011 hasta mediados de 2022, siguiendo la distribución mostrada en la Figura 2.6. Cabe destacar que, entre 2015 y 2020, el promedio de artículos publicados es de 6 por año, lo cual resulta adecuado para un tema tan específico como la codificación de vídeo en entornos de Nube.

A partir de estos trabajos hemos realizado un estudio de los títulos y sus resúmenes para analizar las palabras que aparecen con mayor frecuencia, ob-

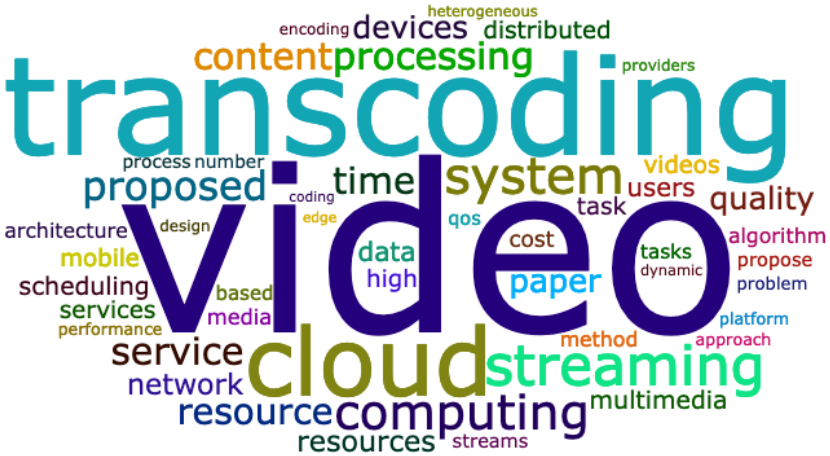


Figura 2.7: Nube de palabras a partir de los resúmenes de los artículos seleccionados.

teniendo la Figura 2.7. En esta figura observamos que los cinco términos con mayor frecuencia son: *video*, *transcoding*, *cloud*, *streaming* y *system*. Todos ellos están ordenados de mayor a menor frecuencia de aparición.

Además, los términos que aparecen en la Figura 2.5 están relacionados con el tema de esta revisión lo que sugiere que la selección de artículos ha sido adecuada.

Los trabajos seleccionados se han agrupado en tres categorías principales según el tipo de tecnología de virtualización utilizada (VMs vs. contenedores) y la entrega de contenido multimedia en la Nube (mostrado en la Figura 2.8).

En el grupo basado en VMs, se han identificado tres subgrupos: clusters estáticos desarrollados con MapReduce u otras tecnologías, infraestructuras elásticas basadas en Nubes públicas, privadas o federadas, y trabajos que utilizan exclusivamente máquinas virtuales.

Luego, se han identificado soluciones basadas en contenedores que se han dividido en dos categorías: los trabajos que utilizan *serverless*, que se basan en el modelo de ejecución en el que el proveedor de la Nube se encarga de ejecutar un fragmento de código asignando dinámicamente los recursos, y otras soluciones implementadas utilizando únicamente contenedores simples.

Por último, se ha incluido un grupo basado en la distribución, que muestra soluciones de transcodificación para mejorar la entrega del flujo de vídeo. En

este caso, se pueden distinguir entre soluciones mediante arquitecturas *Edge* o CDN.

Se puede encontrar una descripción más detallada de cada una de estas tecnologías, así como un resumen de las principales aportaciones de cada uno de los artículos siguiendo esta clasificación, en el trabajo (Moina-Rivera et al., 2023b).

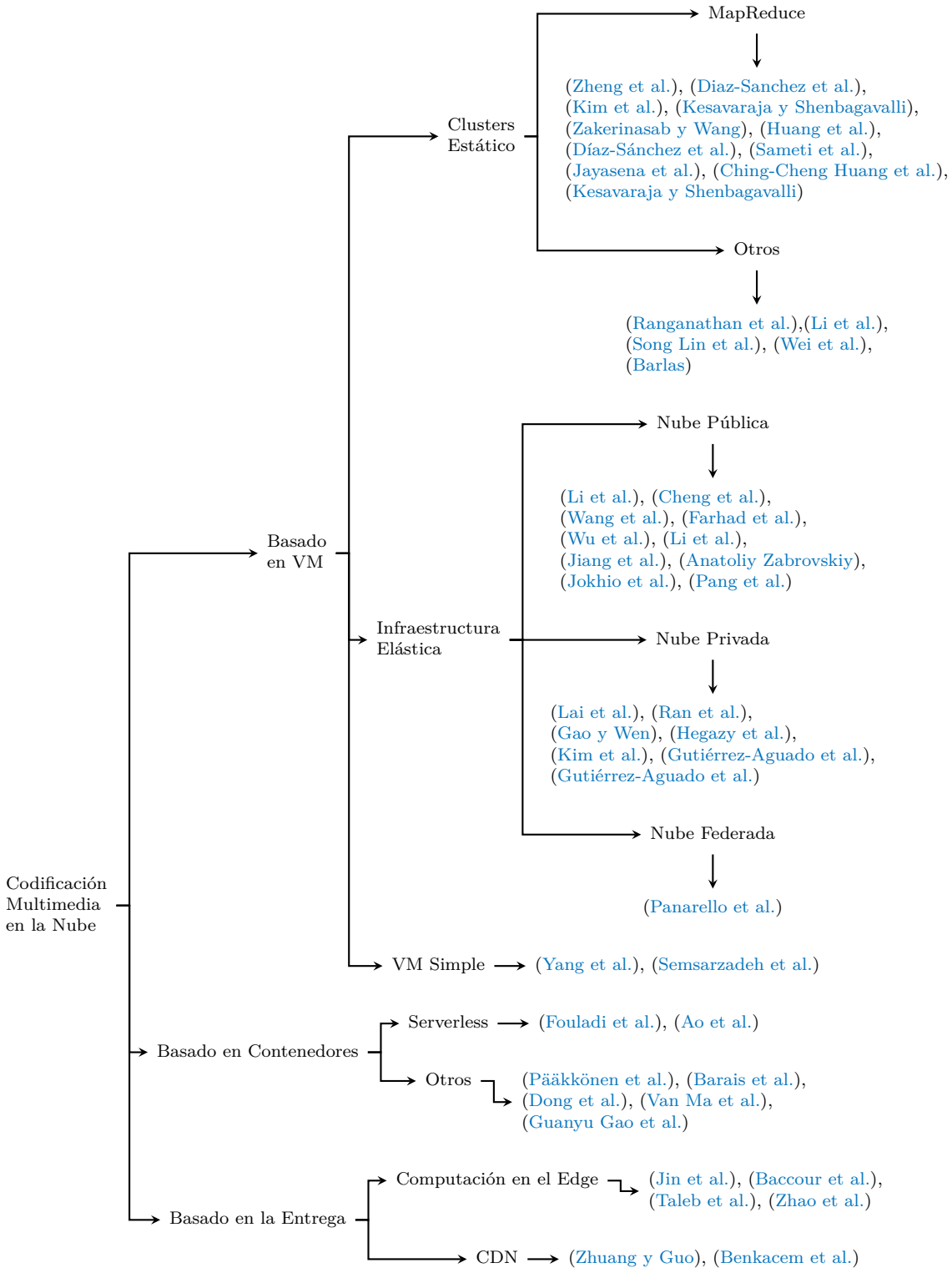


Figura 2.8: Clasificación de los artículos seleccionados.

2.4.2. Análisis de los trabajos según los códecs de vídeo y el método de entrega

2.4.2.1. Clasificación basada en los códecs de vídeo

Como hemos visto en la subsección 2.2.1, existen varios códecs de vídeo que se pueden utilizar en los escenarios multimedia. La Tabla 2.5 muestra el códec de vídeo utilizado por cada trabajo estudiado en esta revisión. Como podemos observar, el códec más utilizado en las propuestas es el códec H264/AVC. El trabajo que más códecs ha utilizado en sus pruebas es (Gutiérrez-Aguado et al., 2020a), donde los autores comparan el comportamiento de su propuesta utilizando H265/HEVC, VP9 y AV1, además es el único que ha utilizado el códec AV1 de última generación. Otra conclusión que podemos extraer de esta tabla es que sólo hay 3 trabajos que han utilizado el códec VP9. Muchos de los trabajos analizados no indican qué códec de vídeo se ha utilizado en sus propuestas, por lo que la tabla indica N/A (Not Available).

2.4.2.2. Clasificación basada en los métodos de transmisión de vídeo en línea

En la actualidad, existen principalmente dos métodos de vídeo *streaming* en línea mediante DASH. Por un lado, el VOD se refiere a un sistema que permite a los usuarios ver contenido de vídeo, como películas o programas de televisión, en cualquier momento que elijan, en lugar de en un horario programado. Con VOD, los usuarios pueden ver contenido de vídeo pregrabado (por ejemplo, contenido producido por una transmisión en directo –Live-to-VOD) en sus propios horarios, a menudo a través de un servicio de suscripción o comprando o alquilando títulos individuales. Por otro lado, la transmisión en directo, también conocida como *live*, se refiere a la transmisión en "tiempo real" de contenido de vídeo y audio a través de Internet. Las transmisiones en vivo pueden ser vistas por cualquier persona con una conexión a Internet y se transmiten típicamente en directo, mientras ocurre el evento. Las transmisiones en vivo pueden incluir una variedad de contenido, como eventos deportivos, conciertos, noticias y más.

Aunque ambos sistemas disponen de una infraestructura para llevar a cabo las tareas de codificación, servicio, *streaming*, monitorización, etc. los sistemas de *live streaming* tienen unos requisitos más estrictos que los sistemas VOD, por ello en este subapartado vamos a realizar una clasificación de

Tabla 2.5: Códecs de vídeo utilizados en cada trabajo, empleados ya sea en la propuesta de su modelo o en la fase de pruebas.

Trabajos	MPEG1/2	H264/AVC	H265/HEVC	VP8	VP9	AV1	EE4
(Li et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Jokhio et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Zheng et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Diaz-Sanchez et al.)	✓	N/A	N/A	N/A	N/A	N/A	N/A
(Kim et al.)	✓	N/A	N/A	N/A	N/A	N/A	N/A
(Kesavaraja y Shenbagavalli)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Zakerinasab y Wang)	N/A	✓/SVC	N/A	N/A	N/A	N/A	N/A
(Huang et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Díaz-Sánchez et al.)	✓	✓/SVC	N/A	N/A	N/A	N/A	N/A
(Ching-Cheng Huang et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Jayasena et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Kesavaraja y Shenbagavalli)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Sameti et al.)	N/A	N/A	✓	N/A	N/A	N/A	N/A
(Barlas)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Song Lin et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Wei et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Pang et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Ranganathan et al.)	N/A	✓	N/A	N/A	✓	N/A	N/A
(Li et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Cheng et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Wang et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Farhad et al.)	✓	✓	N/A	N/A	N/A	N/A	N/A
(Wu et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Li et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Jiang et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Anatoliy Zabrovskiy)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Lai et al.)	N/A	✓/SVC	N/A	N/A	N/A	N/A	N/A
(Ran et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Gao y Wen)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Hegazy et al.)	N/A	N/A	✓	N/A	N/A	N/A	N/A
(Kim et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Gutiérrez-Aguado et al.)	N/A	N/A	✓	N/A	✓	N/A	N/A
(Gutiérrez-Aguado et al.)	N/A	N/A	✓	N/A	✓	✓	N/A
(Panarello et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Yang et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Semsarzadeh et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Fouladi et al.)	N/A	N/A	N/A	✓	N/A	N/A	N/A
(Ao et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Guanyu Gao et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Barais et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Dong et al.)	N/A	N/A	✓	N/A	N/A	N/A	N/A
(Van Ma et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Pääkkönen et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Jin et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Baccour et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Taleb et al.)	N/A	✓	N/A	N/A	N/A	N/A	N/A
(Zhao et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
(Zhuang y Guo)	N/A	N/A	N/A	N/A	N/A	N/A	✓
(Benkacem et al.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A

los trabajos seleccionados en función de si cada sistema está pensado para trabajar en VOD, Live, o no se especifica. Esta información se refleja en la

Tabla 2.6.

Si prestamos atención a los escenarios en directo, las soluciones propuestas deben ofrecer un contenido codificado con un retardo de extremo a extremo inferior a 15 segundos, lo que puede aprovechar la alta capacidad de computación y el grado de paralelismo del entorno de Nube combinado con la computación de borde.

Tabla 2.6: Clasificación de los trabajos en función de los métodos de transmisión.

Tipo de transmisión	**Trabajos**
VOD (asociado al ancho de banda)	(Diaz-Sanchez et al.), (Kim et al.), (Kesavaraja y Shenbagavalli), (Zakerinasab y Wang), (Sameti et al.), (Barlas), (Li et al.), (Song Lin et al.), (Li et al.), (Wang et al.), (Li et al.), (Anatoliy Zabrovskiy), (Gao y Wen), (Kim et al.), (Gutiérrez-Aguado et al.), (Gutiérrez-Aguado et al.), (Panarello et al.), (Guanyu Gao et al.), (Barais et al.), (Van Ma et al.), (Jin et al.), (Baccour et al.), (Zhao et al.), (Zhuang y Guo), (Benkacem et al.)
Live (asociado a retardo y jitter)	(Huang et al.), (Díaz-Sánchez et al.), (Ching-Cheng Huang et al.), (Kesavaraja y Shenbagavalli), (Jokhio et al.), (Wei et al.), (Pang et al.), (Ranganathan et al.), (Cheng et al.), (Farhad et al.), (Wu et al.), (Jiang et al.), (Hegazy et al.), (Yang et al.), (Semsarzadeh et al.), (Fouladi et al.), (Ao et al.), (Pääkkönen et al.), (Taleb et al.)
No especificado	(Zheng et al.), (Jayasena et al.), (Lai et al.), (Ran et al.), Dong et al.)

2.4.3. Relevancia de los trabajos seleccionados

En esta subsección, vamos a analizar la relevancia de cada artículo seleccionado en este estudio. Para analizar la relevancia de cada uno de ellos, nos basaremos en el número de citas extraídas de Google Scholar[4].

Como podemos ver en la Tabla 2.7, los trabajos seleccionados se han agrupado por año de publicación. En el año 2011, solo aparece una publicación, que tiene 5 referencias. En el año siguiente, 2012, aparecen cuatro trabajos con 174 referencias. De los cuatro trabajos publicados ese año, el trabajo (Li et al., 2012) tiene 125 referencias, un número elevado debido al nivel de especialización del tema de esta revisión. En 2013 se publicaron tres trabajos,

[4]Google Scholar Website: https://scholar.google.com/

que han sido referenciados por 227 trabajos. En 2014 se publicaron el mismo número de trabajos que en 2013, pero su impacto fue menor, ya que estos trabajos solo obtuvieron 70 referencias.

En 2015 y 2016, se publicaron 6 y 8 artículos, respectivamente. El número de referencias a los artículos publicados en 2015 es de 134, mientras que en 2016 es de 119. Las publicaciones de 2017 han obtenido el mayor número de referencias (444). (Fouladi et al., 2017), con 274 referencias, es el artículo más citado de esta revisión, duplicando su relevancia con respecto al segundo más relevante (Ao et al., 2018a), publicado en 2018. En ese año, 2018, se publicaron 5 artículos, que tienen 174 referencias.

Por último, se publicaron seis artículos en 2019 y otros seis en 2020. Los trabajos de 2019 tienen 58 referencias y los de 2020 36. En 2021 y 2022 sólo se ha publicado un trabajo cada año, obteniéndose 16 referencias para (Ranganathan et al., 2021) y 5 para el trabajo (Anatoliy Zabrovskiy, 2022). El número total de citas de los 49 trabajos seleccionados para esta revisión supera las 1.460 referencias. Esto es indicativo de la importancia del tema para el mundo científico. Además, con los nuevos paradigmas asociados a la Nube y a los sistemas multimedia que se implantarán en los próximos años, estos indicadores mejorarán sin duda.

Tabla 2.7: Número de citas por año.

Año	Trabajos	Número de citas	Total de citas por año
2011	(Zheng et al.)	5	5
2012	(Diaz-Sanchez et al.)	5	174
	(Barlas)	26	
	(Li et al.))	125	
	(Zhuang y Guo)	18	
2013	(Jokhio et al.)	107	227
	(Song Lin et al.)	32	
	(Lai et al.)	88	
2014	(Kim et al.)	25	70
	(Cheng et al.)	32	
	(Ran et al.)	13	
2015	(Kesavaraja y Shenbagavalli)	2	134
	(Zakerinasab y Wang)	18	
	(Huang et al.)	3	
	(Yang et al.)	5	
	(Semsarzadeh et al.)	21	
	(Jin et al.)	85	
2016	(Díaz-Sánchez et al.)	4	119
	(Ching-Cheng Huang et al.)	4	
	(Li et al.)	51	
	(Wang et al.)	11	
	(Farhad et al.)	6	
	(Gao y Wen)	19	
	(Guanyu Gao et al.)	12	
	(Barais et al.)	12	
2017	(Jayasena et al.)	25	444
	(Wei et al.)	66	
	(Wu et al.)	9	
	(Li et al.)	70	
	(Fouladi et al.)	274	
2018	(Kesavaraja y Shenbagavalli)	7	174
	(Sameti et al.)	7	
	(Benkacem et al.)	17	
	(Dong et al.)	7	
	(Ao et al.)	136	
2019	(Pang et al.)	1	58
	(Jiang et al.)	4	
	(Hegazy et al.)	11	
	(Taleb et al.)	24	
	(Van Ma et al.)	8	
	(Pääkkönen et al.)	10	
2020	(Kim et al.)	4	36
	(Gutiérrez-Aguado et al.)	0	
	(Gutiérrez-Aguado et al.)	3	
	(Panarello et al.)	5	
	(Baccour et al.)	23	
	(Zhao et al.)	1	
2021	(Ranganathan et al.)	16	16
2022	(Anatoliy Zabrovskiy)	5	5
	Total de citas entre 2011-2022		1462

2.5. Trabajos relacionados con las propuestas presentadas en esta tesis

En esta sección, se presentan diversas contribuciones significativas realizadas por la comunidad científica en el ámbito de la preparación y procesamiento de contenido multimedia, enfocándose particularmente en el vídeo. La estructura de esta sección sigue una secuencia lógica, correspondiente al estado del arte en cada uno de los temas abordados en los capítulos posteriores de la presente tesis doctoral.

2.5.1. Aproximaciones a la codificación de segmentos de vídeo en base a un valor objetivo de calidad

Se han propuesto diferentes esquemas de codificación para la preparación de contenidos de vídeo bajo demanda para HAS. Los autores Adzic et al. (2012) muestran un proceso de segmentación basado en el contenido que ofrece un buen equilibrio entre los requisitos de entrega de la red y el nivel de distorsión del vídeo. Esta solución ahorra un 10 % de ancho de banda de media para los mismos niveles de calidad objetivos. Sin embargo, esta propuesta no se ocupa de la codificación basada en la calidad y no aborda aspectos como la codificación multirresolución, el tiempo de codificación y varias métricas de calidad que son importantes para validar la propuesta.

Netflix ha propuesto varios enfoques (véase (Aaron et al., 2015) y (De Cock et al., 2016)) en los que muestran soluciones para codificar vídeos en función de su calidad. En (Aaron et al., 2015), analizan cada vídeo para determinar la receta de codificación óptima en función de su complejidad, pero detectaron en un estudio posterior que esta solución introducía importantes artefactos de codificación. En su siguiente estudio (De Cock et al., 2016), proponen un sistema para controlar la tasa de bits-resolución-calidad por segmento con el fin de lograr una calidad consistente en todos los niveles, dependiendo de la complejidad del vídeo fuente, pero requiere una codificación de dos o tres pasadas por segmento.

En la última propuesta de Netflix (Katsavounidis, 2018), lo que se propone es un sistema de codificación basado en la calidad. Este sistema tiene como entradas: a) el vídeo codificado con múltiples parámetros de cuantificación

fija (QP fijo)[5] para cada resolución y cada toma[6] y b) la distorsión (inversa de VMAF) de cada vídeo codificado. A continuación, se calcula la envolvente convexa de los puntos (R,D) de cada toma. A partir de estos puntos, el sistema selecciona las tomas para producir codificaciones óptimas; finalmente, se combinan para crear una codificación para toda la secuencia de vídeo. El principal problema de esta propuesta es el número de pasos y codificaciones que se requieren para cubrir un rango de bitrate/calidad deseado por toma.

Otro trabajo, en el que sus autores muestran dos enfoques novedosos para lograr una mejor compensación entre la calidad del vídeo y el uso de datos es (Qin et al., 2019). En este caso, el primer enfoque para cada segmento del vídeo, utiliza un filtro que limita la elección del segmento de mayor calidad para la adaptación de la tasa ABR (*Adaptive Bitrate*) al segmento cuya calidad es la más cercana a la calidad objetivo. A continuación, en el segundo enfoque, se diseña un algoritmo holístico de adaptación de tasa de bits en función del ancho de banda estimado de la red para evitar atascos, adaptándose a la calidad deseada para reducir el consumo de ancho de banda, y minimizando el cambio de calidad entre trozos para mejorar la fluidez de la reproducción. Esta propuesta muestra buenos resultados en términos de uso de datos y desviación media de la calidad objetivo, pero no tiene en cuenta los tiempos de procesamiento, un aspecto importante a la hora de pensar en una solución integral. Otro aspecto que no queda claro es la selección de la calidad, ya que el documento sólo menciona tres niveles de calidad: *good*, *better* y *best*.

En (Amirpour et al., 2021), los autores muestran diferentes enfoques para la codificación *multirate* eficiente con respecto al rendimiento de la codificación paralela. Proponen un método mejorado utilizando la representación de calidad media como representación de referencia, que consigue los mejores resultados globales en términos de tiempo-complejidad de codificación, incluyendo mejoras significativas mientras se codifica la representación de mayor calidad. En ese trabajo, la propuesta se prueba utilizando el códec de vídeo estándar HEVC (Sullivan et al., 2012), que tiene una tasa de implantación muy baja debido a sus derechos. La propuesta habla de una codificación basada en calidades, pero no se especifican los valores VMAF y PSNR obje-

[5]El parámetro de cuantificación controla la cantidad de compresión de cada macrobloque en un fotograma. Los valores grandes significan que habrá mayor cuantificación, más compresión y menor calidad (...) varía de 0 a 51 en H.264, y puede configurar fácilmente un QP fijo para todo su proceso de codificación con x264 y x265 (...) libvpx no tiene un modo QP fijo (Robitza, 2017).

[6]Las tomas o escenas son segmentos de vídeo cortos filmados con la misma cámara bajo condiciones ambientales y de iluminación relativamente constante.

tivo y se centra en el rendimiento del codificador HEVC y sus valores QP. Por lo tanto, no está claro el uso de las métricas indicadas en el proceso de codificación.

Por último, otros esquemas de codificación de vídeo basados en la inteligencia artificial están cobrando importancia. Una propuesta relacionada con este concepto es (Xing et al., 2019), en este trabajo se presenta una solución de control de tasa adaptable al contenido. Los autores proponen un modelo basado en redes neuronales capaz de predecir el valor óptimo del factor de tasa para reducir las fluctuaciones de calidad en un proceso de codificación. Para obtener una efectividad de hasta el 77,63 % en la predicción de este parámetro, el modelo necesita, para la extracción y caracterización de fotogramas, un tiempo medio de aproximadamente el 44 % de la duración del segmento de vídeo.

Otro, por ejemplo, es (Covell et al., 2016), donde sus autores tratan de encontrar un *bitrate* objetivo para cada segmento, sin necesidad de codificación multipaso. Para resolver esta cuestión, proponen una red neuronal para predecir los parámetros de un modelo que relaciona el *bitrate* con varias propiedades del vídeo. Su solución puede utilizar la estructura proporcionada por ese modelo para guiar el entrenamiento de su red neuronal y estimar el valor correcto de CRF para una tasa de bits deseada. Sin conocer el modelo de tasa de bits, su sistema no converge. Según las pruebas presentadas, su solución puede estimar el parámetro de control de calidad correcto en el 65 % de sus casos de prueba sin utilizar ningún paso de transcodificación previo. La principal limitación de la propuesta es la falta de validación de los *bitrate* seleccionados para cada prueba con algunas métricas de calidad.

Como resumen de este apartado, podemos ver que hay pocos trabajos que intentan conseguir una codificación basada en la calidad, pero ninguno de ellos introduce las siguientes cuestiones, que se tienen en cuenta en nuestra propuesta que se presenta en el capítulo 3:

- proceso dinámico de codificación basado en la calidad para codificar con una calidad objetivo proporcionada por el usuario,
- tiempos de codificación para validar la eficiencia de nuestro sistema y
- validación del sistema con otras métricas de calidad para comprobar que nuestra propuesta funciona correctamente.

2.5.2. Mejora de la codificación DASH mediante análisis de escenas y técnicas de reducción de escala para la transmisión VOD

En esta subsección, analizamos en primer lugar los trabajos anteriores relacionados con la codificación de segmentos de tamaño no uniforme para DASH y, a continuación, prestamos atención a varios trabajos que utilizan técnicas de reducción de escala para mejorar el proceso de codificación de vídeo.

2.5.2.1. Codificación de segmentos de tamaño variable

A lo largo de los años se han propuesto varios esquemas de codificación para la preparación de contenidos de vídeo bajo demanda en DASH. La mayoría de los trabajos se centran en la codificación de vídeo de segmentos uniformes o de duración fija y en cómo la duración del segmento afecta a la calidad percibida por el usuario final.

En (Lederer, 2015), los autores recomiendan utilizar segmentos de unos 2 a 4 segundos, lo que supone un buen compromiso entre eficiencia de codificación y flexibilidad para la adaptación del flujo a los cambios de ancho de banda. Sin embargo, se ha estudiado la alternativa de utilizar segmentos de duración variable para analizar su impacto en los procesos de codificación y *streaming* de vídeo (Lederer et al., 2012).

Los autores de (Zach y Slanina, 2018b) analizan la tasa de bits utilizando diferentes tamaños de segmentos fijos (2, 5 y 7) y variables (basados en escenas) de duración entre 2 y 7 segundos para *streaming* DASH. En su estudio utilizan dos códecs: AVC y HEVC y cuatro vídeos de alta definición (FHD) (no identificados y no descargables). Los resultados muestran que los segmentos largos producen una reducción de la tasa de bits en comparación con los más cortos. Los segmentos de duración variable proporcionan resultados similares y se reduce la variabilidad de la tasa de bits.

Existen varias diferencias entre este trabajo y nuestra propuesta presentada en la sección 3.2, las principales son: a) utilizamos 10 vídeos UHD para añadir más variabilidad y son de libre acceso para poder recrear las mismas pruebas, b) utilizamos AVC y VP9, ya que HEVC no es soportado por la mayoría de navegadores web, y c) utilizamos tres métricas de calidad: VMAF

(Video Multimethod Assessment Fusion), SSIM (Structural Similarity Index Measure) y PSNR (Peak Signal-to-Noise Ratio) para validar nuestra propuesta.

En (Schwarzmann et al., 2018), se evalúa la duración variable de los segmentos alineados con las características del vídeo. Se estudia la sobrecarga que implica la codificación y, de acuerdo con el enfoque propuesto, el uso de segmentos variables es más eficiente en términos de tasa de bits en comparación con el enfoque con duración de segmento fija, manteniendo al mismo tiempo la calidad. Los mismos autores mejoran su estudio en (Schwarzmann et al., 2020b), donde se realiza una comparación de la codificación fija frente a la variable (basada en escenas) para HAS. Utilizan cuatro vídeos y cinco resoluciones. Realizan dos conjuntos de experimentos: tamaño y calidad; y estimación de la QoE durante el streaming. Codifican los vídeos utilizando x264 a 2160p para encontrar las escenas (permitiendo una duración máxima de 10 segundos). A continuación, codifican los vídeos con diferentes factores de tasa constante (CRF) y los comparan con SSIM. Llegan a la conclusión de que la duración variable de los segmentos proporciona una reducción del tamaño final del vídeo y puede ofrecer una QoE similar o mejor.

Existen varias diferencias entre estos trabajos anteriores y nuestra aportación presentada en la 3.3. En primer lugar, evaluamos el impacto de la resolución en el cálculo de las escenas. Nosotros utilizamos más vídeos y más códecs, estos estudios previos sólo codifican los vídeos utilizando x264, mientras que nosotros utilizaremos también VP9. Además, los autores no proporcionan información sobre el rendimiento del proceso general de codificación, como el uso de CPU, memoria o tiempo de codificación. Utilizaremos VMAF, SSIM y PSNR para validar nuestra propuesta y compararemos nuestra solución con una solución de segmento fijo de 4s con un CRF medio para obtener un nivel de calidad similar.

2.5.2.2. Codificación downscaling

Según (Bruckstein et al., 2003), las técnicas previas de *downscaling* podrían ser una forma adecuada de mejorar la compresión en imágenes y vídeos. En (Dar y Bruckstein, 2014) se muestra un ejemplo que aplica estas técnicas en vídeos. En este trabajo, los autores proponen un método que incluye el *downscaling* antes de la compresión y el correspondiente up-scaling después, sin modificar el códec x264. Su método mejora la eficiencia de la compresión con tasas de bits bajas, pero cuando se requieren vídeos de alta calidad este

método no funciona correctamente.

La principal diferencia entre (Dar y Bruckstein, 2014) y nuestra propuesta de la sección 4.2.1 es que nuestro sistema sólo requiere técnicas de *downscaling* para conocer datos sobre el vídeo, por ejemplo, los cambios de escena. A continuación, los vídeos se codifican a diferentes resoluciones con esta información para mejorar la eficiencia del proceso de codificación.

Por último, el último códec de vídeo LCEVC propuesto por MPEG (Maurer et al., 2020) tiene un proceso de codificación basado en dos "downscalers". Esta salida invoca un códec base, que produce un flujo de bits base. Su flujo de bits base puede mejorarse con dos subcapas de mejora, en las que se utilizan técnicas de escalado ascendente para añadir mejoras de calidad. Este trabajo concluye con una comparación entre LCEVC, H.264/AVC y H.265/HEVC, donde podemos ver que LCEVC ofrece mayor calidad requiriendo menor tasa de bits y con tiempos de codificación y decodificación similares a los códecs base utilizados.

Las principales diferencias entre este trabajo y nuestra propuesta, presentada en el capítulo 4 de esta memoria, son tres: a) nuestra solución puede ser utilizada por cualquier reproductor que soporte DASH; b) nuestro proceso de *downscaling* se utiliza únicamente para extraer información de las escenas de nuestro vídeo y c) el esquema propuesto puede ser utilizado con diferentes códecs.

2.5.3. Uso de plataformas serverless para la codificación y la evaluación de la calidad del vídeo

Recientemente, las soluciones en la Nube propuestas para los servicios de codificación y transcodificación distribuida de vídeo han evolucionado para adaptarse a los requisitos demandados y para mejorar su rendimiento. Las primeras propuestas de codificación distribuida de vídeo sobre los entornos en la Nube se basaban principalmente en el paradigma MapReduce. Así, en Pereira et al. (2010), se propone una arquitectura *Split* & *Merge* para el despliegue de codificación de vídeo sobre infraestructuras de la Nube, utilizando el estándar H.264. La propuesta de este trabajo se basa en dividir el vídeo en varios chunks y distribuir estos chunks en la infraestructura de la Nube para disminuir el tiempo de codificación.

Pero, en estas soluciones basadas en MapReduce y Spark, el número de

codificadores no puede modificarse dinámicamente una vez iniciada la codificación, aunque se utilice programación dinámica, como en (Sameti et al., 2018b). Esta instanciación dinámica del número de codificadores es crucial para los servicios de codificación de vídeo, caracterizados por una alta demanda y por las aplicaciones de *streaming* adaptativas. Para adaptar el número de codificadores a las necesidades de la transmisión de vídeo, como altas resoluciones de vídeo con restricciones de ancho de banda y requisitos de calidad, se han propuesto nuevas soluciones basadas en la Nube.

Gutiérrez-Aguado et al. (2020b,c) proponen una arquitectura distribuida basada en la Nube, ajustada para la codificación de vídeo (utilizando los codificadores HEVC, VP9 y AV1), que se basa en un pool elástico de workers y servidores multimedia para proporcionar tolerancia a fallos. Esta solución permite adaptar los recursos a la carga de trabajo, ofrece alta disponibilidad y proporciona acceso ubicuo. De este modo, se despliega una aplicación distribuida sobre la arquitectura para dividir el proceso de codificación de un vídeo en varios trabajos que pueden asignarse dinámicamente entre el pool elástico de workers.

Sin embargo, en estas soluciones basadas en la Nube, los workers de codificación dinámica utilizados para la codificación de vídeo distribuida son máquinas virtuales (VM) que requieren un tiempo no despreciable para desplegarse y destruirse. Por esta razón, los contenedores Linux, y especialmente los contenedores Docker, han sido una alternativa a las máquinas virtuales para virtualizar los recursos informáticos. Como proceso ligero, los contenedores permiten encapsular aplicaciones con todas sus dependencias y garantizar su ejecución a través de múltiples plataformas, reduciendo el tiempo de despliegue, y obteniendo mayor eficiencia que con las VMs tradicionales (Sharma et al., 2016).

Dong et al. (2018b) presentaron una propuesta para un sistema de procesamiento multimedia basado en contenedores, el cual se dedicaba al servicio de transcodificación de vídeo para los codificadores H.264 y HEVC en una Nube privada. Sin embargo, esta solución presenta una limitación importante: su esquema de equilibrio de carga es relativamente simple.

Por otro lado, Pääkkönen et al. (2018), desarrollaron una arquitectura con el objetivo de predecir el rendimiento de la transcodificación de vídeo en directo en una plataforma que utiliza Docker. Esta propuesta se centra en la gestión de recursos en la Nube para la transcodificación de vídeo en directo, e incorpora una estrategia de predicción en vivo para optimizar la asignación de recursos en la programación de contenedores de transcodificación. Sin

embargo, cabe mencionar que esta solución se limita al uso de la codificación H.264.

Sameti et al. (2020) proponen CONTRAST, un marco de transcodificación distribuida basado en contenedores para el *streaming* de vídeo interactivo. Esta solución, centrada en la codificación HEVC, utiliza un analizador de perfiles externo para determinar automáticamente el grado de paralelismo y lanza contenedores configurados con el número necesario de núcleos para realizar la transcodificación. Así, se lanzará un contenedor por cada trozo de vídeo con el número de núcleos determinado en la etapa análisis. Esta solución se ha implementado y probado en una Nube pública (Amazon EC2), pero no se proporcionan detalles de implementación.

El uso de contenedores en un clúster añade complejidad debido a la orquestación y programación. Por esta razón, las plataformas de orquestación de contenedores (COP) como Docker Swarm, Nomad o Kubernetes se utilizan en entornos de computación en Nube (Marathe et al., 2019), pero Kubernetes es actualmente el COP estándar "de facto", seguido por OpenShift (que a menudo se conoce como "Enterprise Kubernetes") (Moravcik y Kontsek, 2020).

Sin embargo, luego aparece Serverless, una nueva abstracción de computación en la Nube que facilita la escritura de servicios web robustos y a gran escala (Jangda et al., 2019) que abre una nueva era en la computación en la Nube. Un concepto de gestión de recursos informáticos en el que despliegas las funciones en la plataforma del proveedor de Nube y pagas por el tiempo de ejecución de las funciones cuando son invocadas, todo lo demás es responsabilidad del proveedor de la Nube. Amazon Web Services introdujo en 2014 la primera plataforma sin servidor, AWS Lambda, y abstracciones similares están ahora disponibles en la mayoría de las plataformas de computación en Nube (por ejemplo, Google Cloud Functions, IBM OpenWhisk y Azure Functions) (Martins et al., 2020).

Por ejemplo, un trabajo reciente de (Risco et al., 2021) muestra las ventajas del uso de infraestructuras serverless on-premises como OSCAR, que se basa en OpenFaaS, combinado con el uso de las Nubes serverless públicas como AWS Lambda. En estos casos, la solución on-premise puede ser una mejor solución cuando se requiere una distribución minimalista de Kubernetes y una rápida funcionalidad event-driven, como en la frontera de computación.

Fouladi et al. (2017) proponen un codificador de vídeo (para formatos de vídeo VP8) pensado para el paralelismo de grano fino, utilizando un estilo de

programación funcional, logra el paralelismo de grano fino utilizando AWS Lambda (Wang et al., 2018) y utilizando el framework Mu propuesto. Así, en lugar de invocar a los workers en respuesta a un solo evento, los invocan en masa, miles a la vez. En este enfoque, el codificador VP8 (el único utilizado), tiene que ser modificado para recodificar chunks consecutivos de 6 fotogramas (esta es una elección particular de los autores) para realizar la fase "*rebase*" (ejecutándose en serie) y para concatenar los chunks en la secuencia final de vídeo codificada. Esta implementación podría mejorarse aumentando el tamaño de los chunks (por ejemplo, chunks de 2 segundos o 48 fotogramas) para evitar la comunicación entre los workers (fase "*rebase*") y aumentar significativamente el rendimiento declarado (8 veces más rápido).

Sprocket (Ao et al., 2018b) es un sistema *serverless* implementado en AWS Lambda. Este marco de procesamiento de vídeo permite a los desarrolladores programar una serie de operaciones sobre el contenido de vídeo de forma modular y extensible. Por lo tanto, maneja el acceso subyacente, la codificación y decodificación, y el procesamiento de contenido de vídeo e imágenes a través de operaciones altamente paralelas. Este trabajo amplía el framework Mu (Fouladi et al., 2017) para soportar la creación dinámica de tareas durante el procesamiento, en lugar de la invocación *batch-style* de Lambda, y la mensajería asíncrona y la transferencia de datos entre los workers Lambda. No se proporcionan detalles sobre los codificadores utilizados.

Actualmente, hay varias plataformas *serverless* de código abierto muy populares (es decir, Knative, Kubeless, Nuclio y OpenFaaS) (Li et al., 2021), que son alternativas a los servicios *serverless* ofrecidos por los proveedores de Nube pública y ofrecen nuevas características interesantes.

Estas plataformas presentan diferencias de rendimiento entre los distintos modos de exportación de servicios y autoescalado. Se necesitan más estudios para decidir qué plataforma es mejor, pero Knative y OpenFaaS son muy prometedoras. En particular, Knative tiene muchas características útiles, como la escala a cero y múltiples modos de auto-escalado, y una comunidad activa que podría proporcionar ayuda a los usuarios y desarrolladores.

Como hemos visto en esta subsección, y según nuestro conocimiento, hay pocos trabajos ((Fouladi et al., 2017) y (Ao et al., 2018b)) en los que se hayan desarrollado sistemas de codificación de vídeo en plataformas de Nube con *serverless*. En el capítulo 5, presentaremos nuestra propuesta y la evaluaremos para validar su correcto funcionamiento. En la sección 5.4, se compara nuestra propuesta con los trabajos previos relacionados en la codificación de vídeo sobre plataformas *serverless* para evaluar sus diferencias y similitudes.

2.5.4. Video Quality Metrics Toolkit: Un software de código abierto para evaluar la calidad del vídeo.

Existen numerosas herramientas disponibles para evaluar la calidad del vídeo, incluyendo soluciones comerciales y de código abierto. Con el objetivo de identificar las más relevantes y establecer diferencias con nuestra propuesta, descrita en el capítulo 6, se realizó una evaluación de varias aplicaciones utilizando los siguientes criterios: (i) implementación de al menos cinco métricas objetivas para evaluar la calidad del vídeo, (ii) ausencia de limitaciones en el número de fotogramas analizados, y (iii) posibilidad de instalación en un entorno local. Además, se eligieron aquellas herramientas que resultaron pertinentes y valiosas de entre los 40 primeros resultados de búsqueda, empleando la cadena de búsqueda *("medición de la calidad de vídeo" OR "evaluación de la calidad de vídeo" OR "evaluación objetiva de la calidad de vídeo") AND ("tool" OR "software" OR "program")* en el motor de búsqueda de Google.

A partir de los resultados de la búsqueda, se han evaluado versiones con licencia comercial, gratuitas o de prueba. Estas versiones de prueba oscilan entre 14 y 40 días con sus respectivas limitaciones de uso.

El MSU VQMT Free[7] (versión 14.1), desarrollada por el MSU Graphics & Media Lab Video Group, permite analizar la calidad de vídeo/imagen utilizando métricas de referencia y sin referencia. Esta herramienta implementa las métricas enumeradas en la Tabla 6.1, donde también se puede encontrar un resumen de sus principales características y componentes. La versión gratuita de esta herramienta sólo está disponible para el sistema operativo Windows, mientras que las características de las versiones de pago están disponibles en su página web oficial.

Otra herramienta es VQ Probe Free[8] (versión 13.1) desarrollada por ViCueSoft, que se presenta como una herramienta visual para la comprobación objetiva y subjetiva de la calidad de vídeo. La herramienta implementa las métricas de calidad enumeradas en la Tabla 6.1, junto con un resumen de sus principales características y componentes. Sin embargo, al igual que VQMT Free, esta versión gratuita sólo está disponible para el sistema operativo Windows y con algunas limitaciones.

[7] https://compression.ru/video/quality_measure/video_measurement_tool.html

[8] https://portal.vicuesoft.com

AccepTV presenta otra propuesta con su herramienta Analizador de calidad de vídeo[9] (versión 4.7.5), que es una solución de referencia completa que requiere acceso a ambos vídeos para realizar una evaluación de la calidad del vídeo. La versión de prueba de 14 días ofrece acceso ilimitado a las métricas de calidad enumeradas en la Tabla 6.1, junto con un resumen de sus principales características y componentes. Sin embargo, esta herramienta sólo es compatible con el sistema operativo Windows, a diferencia de las versiones de pago o de prueba de VQMT y VQ Probe.

Además, la herramienta Video Quality Estimator (versión 2022) desarrollada por Elecard, ofrece acceso a una prueba de evaluación de 30 días compatible con los sistemas operativos Linux, Windows y macOS. Sin embargo, está limitada a un número determinado de fotogramas (500 fotogramas), lo que la hace inadecuada para este estudio.

Cabe mencionar que las páginas web oficiales de cada una de las herramientas mencionadas se describen los requisitos mínimos de hardware para su instalación, así como documentación detallada sobre sus características y modos de uso.

En la subsección que sigue, se expondrán las características principales de las métricas integradas en la herramienta `vqmtk` propuesta. Además, se proporcionan las referencias a los proyectos que implementan estas métricas y se indican los valores aproximados requeridos para que la mayoría de ellas alcancen un valor específico del MOS.

Las métricas se han clasificado según los datos de referencia utilizados para calcular la calidad del vídeo, ver Figura 2.9.

2.5.4.1. Métricas objetivas de referencia completa

Peak Signal-to-Noise Ratio (PSNR) (Huynh-Thu y Ghanbari, 2010) es una métrica tradicional de calidad de la imagen, expresada en decibelios. Está basada en el error cuadrático medio (MSE) o en su raíz cuadrada (RMSE), y se utiliza comúnmente para evaluar la calidad de imágenes y vídeos comprimidos. Aunque es una métrica ampliamente utilizada, tiene limitaciones en cuanto a la correlación con la percepción humana de la calidad. A continuación, se muestran las ecuaciones para calcular MSE y PSNR:

[9] https://www.acceptv.com/en/products_vqa.php

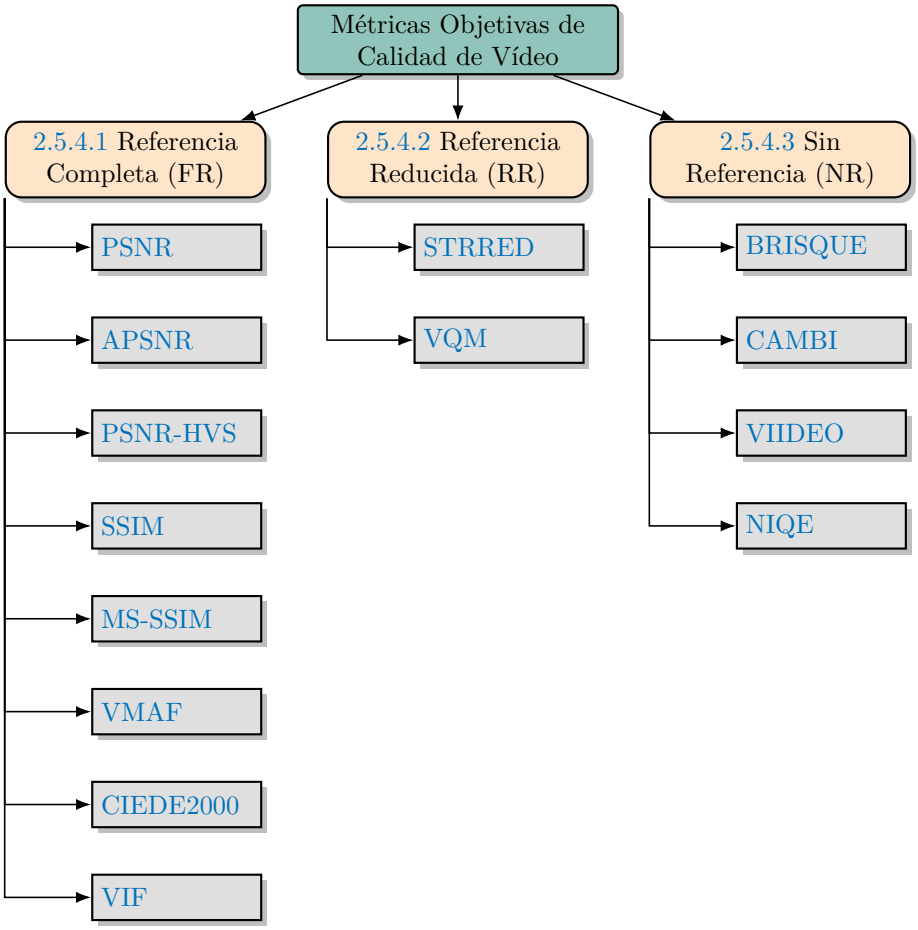


Figura 2.9: Métricas objetivas de calidad de vídeo incluidas en `VQMTK`.

$$\text{MSE} = \frac{1}{W \times H \times N} \sum_{t=1}^{T} \sum_{w=1}^{W} \sum_{h=1}^{H} (O_{t,w,h} - R_{t,w,h})^2 \tag{2.1}$$

$$\text{PSNR} = 10 \log_{10} \left(\frac{2^n - 1}{\text{MSE}} \right) \tag{2.2}$$

En estas ecuaciones, W y H representan el ancho y alto de la imagen, respectivamente, mientras que N es el número total de imágenes en la secuencia de vídeo. T es el índice de tiempo, y O y R representan los valores de los píxeles de la imagen original y la imagen reconstruida, respectivamente. El parámetro n indica la cantidad de bits por píxel.

Para calcular esta métrica de calidad se puede utilizar la biblioteca

`libavfilter` integrada en la herramienta FFmpeg[10].

Según el estudio realizado por (Moldovan et al., 2016) para alcanzar un MOS igual a 5 (Excelente) el valor PSNR debe ser mayor o igual a 36dB.

Average Peak Signal-to-Noise Ratio (APSNR) (Syahbana et al., 2011) se basa en calcular la media aritmética del MSE promedio de cada fotograma, y la PSNR total resulta de la media aritmética de todos estos valores. Esta métrica ofrece una perspectiva diferente al considerar la variación en la calidad del vídeo a lo largo de toda la secuencia.

Para calcular esta métrica de calidad se puede emplear el proyecto de código abierto `av-metrics`[11] en su versión 0.8.1.

Dado que APSNR es otra forma de calcular el promedio de PSNR, siguiendo la recomendación de (Moldovan et al., 2016), para alcanzar un MOS igual a 5 (Excelente), el valor de PSNR debe ser mayor o igual a 36 dB.

PSNR-HVS (Egiazarian et al., 2006) es una extensión del PSNR, la cual incorpora características del sistema visual humano (HVS) para mejorar la correlación con la percepción de calidad. La métrica realiza una transformación DCT en bloques de 8x8 de la imagen, aplica ponderaciones a los coeficientes y, posteriormente, calcula la PSNR de estos coeficientes ponderados. Las ponderaciones utilizadas se basan en el trabajo de investigación publicado por (Ponomarenko et al., 2007).

Para calcular esta métrica de calidad en el marco de la herramienta propuesta, se emplea el proyecto de código abierto `av-metrics`[11].

De acuerdo con el estudio llevado a cabo por (Moldovan et al., 2016), para alcanzar un MOS igual a 4 (Bueno), el valor de PSNR-HVS debe ser mayor o igual a 32 dB.

Structural Similarity Image Metric (SSIM) (Wang et al., 2004) compara dos imágenes, $\mathbf{x}$ y $\mathbf{y}$, utilizando una ventana deslizante de píxeles vecinos. Esta ventana se desplaza desde la esquina superior izquierda hasta la

[10] `https://ffmpeg.org`
[11] `https://docs.rs/crate/av-metrics/0.8.1`

esquina inferior derecha de la imagen, generando un mapa de calidad para la imagen evaluada. La ecuación SSIM se basa en tres componentes que representan la luminancia, el contraste y la estructura de las imágenes.

La ecuación de SSIM es la siguiente:

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{2.3}$$

Donde:

- μ_x y μ_y son las medias de las intensidades de los píxeles de las imágenes $\mathbf{x}$ e $\mathbf{y}$, respectivamente.
- σ_x^2 y σ_y^2 son las varianzas de las intensidades de los píxeles de las imágenes $\mathbf{x}$ e $\mathbf{y}$, respectivamente.
- σ_{xy} es la covarianza entre las intensidades de los píxeles de las imágenes $\mathbf{x}$ e $\mathbf{y}$.
- C_1 y C_2 son constantes que se añaden para evitar la división por cero. Generalmente, se definen como $C_1 = (k_1 L)^2$ y $C_2 = (k_2 L)^2$, donde L es el rango dinámico de las intensidades de píxeles (por ejemplo, 255 para imágenes de 8 bits) y k_1 y k_2 son constantes pequeñas[12].

La ecuación SSIM mide la similitud estructural de las imágenes evaluadas, proporcionando un mapa de calidad que se promedia para obtener un valor global de similitud. Los valores de SSIM oscilan entre 0 y 1, siendo 1 el valor óptimo que indica que las imágenes son idénticas sin ninguna distorsión.

Para calcular esta métrica de calidad una opción es utilizar la biblioteca `libavfilter`, integrada en la herramienta FFmpeg[10].

Según el estudio llevado a cabo por (Moldovan et al., 2016), para alcanzar un MOS igual a 5 (Excelente), el valor de SSIM debe ser mayor o igual a 0.93.

[12]En general, se recomienda utilizar los valores estándar de $k_1 = 0{,}01$ y $k_2 = 0{,}03$, ya que estos han sido probados y validados en diversos estudios y proporcionan un buen equilibrio entre la estabilidad y sensibilidad de la métrica SSIM.

Multi-Scale Structural Similarity (MS-SSIM) es una extensión del SSIM. Este índice de calidad se calcula combinando la comparación de la luminancia, la medida de la distorsión del contraste y la comparación de la estructura en diferentes escalas, de acuerdo con la siguiente ecuación (Wang et al., 2004):

$$\text{MS-SSIM}(x,y) = [l_M(x,y)]^{\alpha_M} \prod_{i=1}^{M} [c_i(x,y))]^{\beta_i} [s_i(x,y))]^{\gamma_i} \tag{2.4}$$

La función MS-SSIM evalúa la similitud estructural entre dos imágenes a múltiples escalas, lo que la hace más robusta y adecuada para imágenes con variaciones de resolución y detalles finos. La ecuación combina la luminancia l_M, el contraste c_i y la estructura s_i en diferentes niveles M de la imagen, ponderando cada componente con coeficientes α_M, β_i y γ_i, respectivamente.

Para calcular esta métrica de calidad se puede utilizar la biblioteca `libvmaf`[13] integrada en la herramienta FFmpeg[10].

Según el estudio realizado por (Moldovan et al., 2016) para alcanzar un MOS de 5 (Excelente), el valor MS-SSIM debe ser mayor o igual a 0.98.

Video Multimethod Assessment Fusion (VMAF) (Li et al., 2016b; Daede et al., 2017) en desarrollo por Netflix es una métrica que pretende aproximarse a la calidad de vídeo percibida por un humano. Esta métrica se basa en la combinación de múltiples métricas de calidad de imagen y vídeo en un solo indicador de calidad percibida. Estas métricas elementales se combinan utilizando una máquina de vectores de soporte (SVM) entrenada con un conjunto de datos de vídeos de referencia y sus versiones comprimidas y/o recaladas. El conjunto de datos de entrenamiento incluye también las puntuaciones de calidad subjetiva obtenidas a través de pruebas realizadas con observadores humanos.

Las métricas elementales utilizadas en VMAF son las siguientes:

1. ANSNR (Anti-noise SNR): Esta métrica es una variante del PSNR que intenta abordar las limitaciones del PSNR al considerar las características del sistema visual humano.

[13] https://github.com/Netflix/vmaf

2. DLM (Detail Loss Measure): Esta métrica evalúa la pérdida de detalles finos en la imagen o el vídeo comprimido en comparación con el original. El DLM se basa en la diferencia de energía entre el contenido original y el comprimido en el dominio de la frecuencia.

3. VIF (Visual Information Fidelity): Esta métrica evalúa la fidelidad de la información visual en la imagen o el vídeo comprimido en relación con el original. VIF tiene en cuenta las propiedades del sistema visual humano, como la sensibilidad a diferentes frecuencias y direcciones espaciales.

4. MCPD (Mean Co-located Pixel Difference): Esta métrica evalúa la diferencia promedio de píxeles co-ubicados entre un fotograma y el fotograma anterior. Es una medida de la distorsión temporal en el vídeo.

VMAF también incluye un proceso de pooling espacial y temporal para agregar las puntuaciones de las métricas elementales en un solo valor de calidad. La métrica VMAF resultante es una puntuación que oscila entre 0 y 100, donde 100 indica la mejor calidad percibida.

Es importante tener en cuenta que VMAF es específico para el conjunto de datos y el modelo de regresión utilizado en su entrenamiento. Por lo tanto, su rendimiento puede variar en función del tipo de contenido y las condiciones de prueba. Sin embargo, un aspecto que distingue a VMAF de otras métricas tradicionales como PSNR o SSIM, según (Li et al., 2020b), es su capacidad para predecir de manera más consistente a través de resoluciones espaciales, tomas y géneros (por ejemplo, animación frente a documental).

Para calcular esta métrica de calidad se puede utilizar la biblioteca `libvmaf`[13] integrada junto a la herramienta FFmpeg[10].

Según el estudio realizado por (Nandakumar et al., 2019), para obtener un MOS superior a 4 (Bueno), el valor VMAF debe ser mayor o igual a 85.

Color Image Quality Assessment Based on CIEDE2000 (Yang et al., 2012)(Luo et al., 2001)(CId, 2004) es una métrica basada en las distancias de color CIEDE, diseñada para cuantificar con precisión las diferencias de color percibidas por el sistema visual humano. Genera una única puntuación considerando la diferencia de luminosidad, la diferencia de tono y la diferencia de croma entre dos colores.

La diferencia de color CIEDE2000 (ΔE) se calcula utilizando la siguiente ecuación:

$$\Delta E = \sqrt{\left(\frac{\Delta L}{K_L S_L}\right)^2 + \left(\frac{\Delta C}{K_C S_C}\right)^2 + \left(\frac{\Delta H}{K_H S_H}\right)^2 + R_T \left(\frac{\Delta C}{K_C S_C}\right)^2 \left(\frac{\Delta H}{K_H S_H}\right)^2} \tag{2.5}$$

Esta ecuación incorpora varios factores de ponderación y parámetros de ajuste, como K_L, S_L, K_C, S_C, K_H, S_H, y R_T, que tienen en cuenta las características del sistema visual humano y mejoran la precisión de la métrica.

El código fuente original de CIEDE2000 está disponible en MATLAB[14].

Para calcular esta métrica de calidad como parte de la herramienta, se utiliza el proyecto público av-metrics[11] en su versión 0.8.1.

En (Yang et al., 2012), se utilizan el coeficiente de correlación lineal (CC) y el error medio absoluto (MAE) para comparar cuantitativamente los resultados de la evaluación de calidad CIEDE2000 con respecto a PSNR y SSIM. Los autores encontraron que CIEDE2000 ofrece una mejor correlación con la percepción humana de las diferencias de color en comparación con PSNR y SSIM, lo que la convierte en una métrica de calidad de imagen más adecuada para evaluar las diferencias de color en imágenes y vídeos.

Visual Information Fidelity (VIF) (Sheikh y Bovik, 2006) mide la pérdida de fidelidad de la información. VIF cuantifica la fidelidad de la información en una imagen de referencia y evalúa cuánta de esta información puede extraerse de una imagen distorsionada. Se trata de un modelo basado en el modelo estadístico de la escena natural (NSS), el modelo de distorsión de la imagen y la "información que el cerebro podría extraer idealmente del HVS".

Para calcular esta métrica de calidad se puede utilizar la librería `libvmaf`[13] integrada en la herramienta FFmpeg[10].

Según el estudio realizado por (Deriche et al., 2010) para alcanzar un MOS de 4 (Bueno) el valor VIF debe ser mayor o igual a 1.

[14] http://www2.ece.rochester.edu/~gsharma/ciede2000/

2.5.4.2. Métricas objetivas de referencia reducida

Spatio-Temporal Reduced Reference Entropic Differencing (STRRED) (Soundararajan y Bovik, 2013) emplea el modelo de mezcla de escalas gaussianas (GSM) para calcular las diferencias de entropía condicional reducida (RRED) entre el vídeo de referencia y el vídeo distorsionado. Esto resulta en índices de RRED temporales (TRRED) y espaciales (SRRED), que tienen como objetivo medir la diferencia de información de movimiento entre los vídeos de referencia y los distorsionados.

El índice STRRED se obtiene mediante el producto de los índices SRRED y TRRED, como se muestra en la siguiente ecuación:

$$\text{STRRED}k = \text{SRRED}k\text{TRRED}_k. \tag{2.6}$$

Para calcular esta métrica, se puede utilizar la API `scikit-video`[15] en su versión 1.1.11.

De acuerdo con el estudio de evaluación subjetiva (Soundararajan y Bovik, 2013), para lograr valores de MOS superiores a 4 (Bueno), los valores de STRRED deben ser inferiores a 13.5. Es importante mencionar que, en este caso, un valor menor de STRRED indica una mayor calidad en términos de preservación de la información de movimiento en el vídeo distorsionado en comparación con el vídeo de referencia.

Video Quality Model (VQM) (Pinson y Wolf, 2004) es una métrica desarrollada por la National Telecommunications and Information Administration (NTIA) y el Institute of Telecommunication Sciences (ITS) que proporciona una medida objetiva de la calidad de vídeo percibida. VQM evalúa los efectos perceptivos de las deficiencias de vídeo, como el desenfoque, el movimiento no natural, el ruido global, la distorsión por bloques y la distorsión cromática, y los combina en una única métrica.

La métrica VQM consiste en una combinación lineal de los siguientes siete parámetros, extraídos de las regiones espacio-temporales (SI-TI) de la secuencia de vídeo:

[15] `http://www.scikit-video.org`

$$\begin{aligned} \text{VQM} = & -0{,}2097 * \mathbf{siloss} \\ & + 0{,}5969 * \mathbf{hv}_{\mathbf{loss}} \\ & + 0{,}2483 * \mathbf{hv}_{\mathbf{gain}} \\ & + 0{,}0192 * \mathbf{chroma}_{\mathbf{spread}} \\ & - 2{,}3416 * \mathbf{si}_{\mathbf{gain}} \\ & + 0{,}0431 * \mathbf{ct}_{\mathbf{atigain}} \\ & + 0{,}0076 * \mathbf{chroma}_{\mathbf{extreme}} \end{aligned} \tag{2.7}$$

Para calcular esta métrica de calidad, una opción es el proyecto público `vqmetric`[16].

En esta implementación, el valor de VQM oscila entre 0 y 1, donde 1 indica la mejor calidad de vídeo y 0 la peor calidad de vídeo.

Según la validación realizada por (Pinson y Wolf, 2004), para alcanzar un MOS superior a 4 (Bueno), el valor de VQM debe ser superior a 0.8. Esto implica que, a medida que el valor de VQM se acerca a 1, la calidad del vídeo percibido mejora significativamente.

2.5.4.3. Métricas objetivas sin referencia

Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) (Mittal et al., 2011) utiliza estadísticas de escena a partir de los coeficientes de luminancia normalizados localmente para cuantificar posibles pérdidas de naturalidad en la imagen, utilizando NSS (Natural Scene Statistics) para evaluar la calidad de la imagen en el dominio espacial.

El principio en el que se basa el diseño de las características BRISQUE es que las imágenes naturales obedecen a propiedades estadísticas regulares específicas, que se ven alteradas por la presencia de distorsiones (Mittal et al., 2012). Las características BRISQUE se calcularon en varias escalas. Con un total de 36 características, 18 en cada escala. Además, las características también utilizan un modelo para los productos de pares de valores de luminancia vecinos (normalizados).

Para calcular esta métrica se puede utilizar la API `scikit-video`[15] en su versión 1.1.11.

[16] https://github.com/grishnagkh/vqmetric

Según el estudio de (Vranješ et al., 2019), para alcanzar valores DMOS equivalentes a 0.8, los valores de BRISQUE deben ser superiores a 70.

Contrast-aware Multiscale Banding Index (CAMBI) (Tandon et al., 2021) es un detector de artefactos de bandas desarrollado por Netflix con el objetivo de mejorar aún más la calidad del vídeo entregado. Actúa como un detector de bandas sin referencia, tomando un vídeo distorsionado como entrada y generando una puntuación de visibilidad de bandas como salida. El algoritmo extrae mapas a nivel de píxel en múltiples escalas para los fotogramas del vídeo codificado y combina estos mapas en un único índice basado en la función CSF (human contrast sensitivity). CAMBI es temporalmente estable dentro de una misma toma de vídeo y logra una correlación superior a 0.94 tanto en términos de Coeficiente de Correlación Lineal de Pearson (PLCC) como de Coeficiente de Correlación por Orden de Rango de Spearman (SROCC).

En el futuro, el equipo de Netflix planea integrar CAMBI en VMAF como una de las características, permitiendo una evaluación precisa de la calidad del vídeo en presencia de bandas y otros artefactos (Afonso et al., 2022).

Para calcular esta métrica de calidad se utiliza la librería `libvmaf`[13] integrada en la herramienta FFmpeg[10].

Según el estudio realizado por (Tandon et al., 2021), para alcanzar un MOS de 4 (Bueno), el valor CAMBI debe ser igual o inferior a 8.

Video Intrinsic Integrity and Distortion Evaluation Oracle (VIIDEO) (Mittal et al., 2016) es una métrica de calidad de vídeo sin referencia, lo que significa que no requiere valoraciones humanas sobre la calidad del vídeo. Se basa en modelos de regularidades estadísticas intrínsecas observadas en vídeos naturales, que se utilizan para cuantificar las perturbaciones introducidas por las distorsiones. De este modo, un algoritmo derivado del modelo VIIDEO es capaz de predecir la calidad de los vídeos distorsionados sin necesidad de información externa sobre la fuente prístina, las distorsiones previstas o las opiniones humanas sobre la calidad del vídeo.

Para calcular esta métrica, se utiliza la API `scikit-video`[15] en su versión 1.1.11.

Según el estudio realizado por (Sinno y Bovik, 2019), se puede alcanzar

una puntuación de calidad regular con valores de VIIDEO superiores a 0.4. Esto implica que, a medida que el valor de VIIDEO se acerca o supera 0.4, la calidad del vídeo se considera aceptable o mejor en términos de percepción humana.

Naturalness Image Quality Evaluator (NIQE) (Mittal et al., 2013) a diferencia de otros métodos como BRISQUE, esta métrica no requiere datos de entrenamiento con evaluaciones humanas subjetivas. En su lugar, el modelo se construye a partir de características extraídas de un corpus de imágenes naturales no distorsionadas. Esto lo convierte en el primer algoritmo de Evaluación de Calidad de Imagen sin Referencia (NR-IQA) verdaderamente independiente de la distorsión. El modelo se construyó utilizando un conjunto amplio de imágenes prístinas, es decir, sin distorsión.

Para calcular esta métrica, se utiliza la API `scikit-video`[15] en su versión 1.1.11.

Según un estudio realizado por Eerola et al. (2014), los valores de NIQE inferiores a 15 permiten obtener una puntuación de buena calidad en términos de percepción humana de la imagen. Esto sugiere que, a medida que el valor de NIQE disminuye, la calidad de la imagen se considera más cercana a una imagen natural y sin distorsiones.

2.6. Conclusiones

En los últimos años, el tráfico de red ha sido dominado por la transmisión de vídeo. La primera parte de este capítulo ofrece una revisión exhaustiva de la codificación de vídeo en la Nube, abordando la evolución de la transmisión y destacando la importancia de los sistemas en la Nube en entornos multimedia. Para llevar a cabo este análisis, se aplicó la técnica de revisión sistemática de la literatura (SLR), lo que permitió identificar 49 documentos relevantes. Estos trabajos fueron clasificados según el tipo de tecnología de virtualización (VM frente a contenedores) y la entrega de contenido multimedia en la Nube. Además, se realizó un estudio detallado de sus metadatos.

Posteriormente, se presentan las principales contribuciones de los trabajos relacionados con la codificación de segmentos de vídeo en función de la calidad, la mejora de la codificación DASH mediante análisis de escenas y

técnicas de reducción de escala, la codificación de segmentos con tamaño variable y la codificación de vídeo en la Nube utilizando funciones *serverless*. Por último, se expone un conjunto de herramientas empleadas para calcular la calidad de vídeo, junto con la descripción de las principales métricas de calidad utilizadas en este ámbito. Esta revisión proporciona una visión amplia y actualizada del estado del arte en la codificación de vídeo en la Nube, así como una base sólida para futuras investigaciones y desarrollos en este campo, que está en constante evolución.

Capítulo 3

Aproximaciones a la codificación de segmentos de vídeo en base a un valor objetivo de calidad

La ciencia es todo aquello sobre lo cual siempre cabe discusión.

José Ortega y Gasset

Resumen:

Hoy en día, la mayoría de las plataformas de *streaming* utilizan *HTTP Adaptive Streaming* para transmitir el contenido multimedia a los usuarios finales con el fin de garantizar una buena calidad de experiencia. Sin embargo, esta QoE debe estar garantizada desde el vídeo que se va a transmitir, es decir, el vídeo debe tener una calidad adecuada con el valor más bajo posible del *bitrate* antes de la transmisión. Para resolver esta cuestión, en este capítulo, presentamos el desarrollo y la implementación de dos aproximaciones para la codificación de segmentos de vídeo para una transmisión de VOD para HAS, concretamente sobre el estándar DASH. En los enfoques propuestos, para crear una representación DASH, el vídeo de entrada de alta resolución se divide en segmentos temporales, y para cada segmento se obtiene el valor del CRF para codificar ese segmento de forma que se alcance el valor de la métrica de calidad deseado. Las propuestas se han validado con vídeos 4K y diferentes tamaños de segmento, y se

ha comparado con un esquema de codificación de CRF fijo (el mismo valor de CRF para todos los segmentos). Los resultados muestran que las soluciones propuestas pueden reducir el tamaño del vídeo manteniendo la misma calidad. El desarrollo de este capítulo está respaldado por las contribuciones (Moina-Rivera et al., 2020), (Moina-Rivera et al., 2021b) y (Moina-Rivera et al., 2021a).

3.1. Introducción

Para la mayoría de las plataformas de *streaming* actuales, HTTP Adaptive Streaming o HAS, como mencionamos en la introducción de este documento, es el estándar de *facto* para la transmisión de vídeo en la industria. En HAS, un vídeo se divide en una serie de segmentos temporales —en el orden de segundos—, y cada segmento se codifica a varias resoluciones y tasas de bits, para crear así varias representaciones del segmento de vídeo con diferentes niveles de calidad. La información multimedia, que incluye las representaciones disponibles, la duración de los segmentos y los URI, se describe en un fichero de metadatos denominado archivo manifiesto. Cada cliente ejecuta un algoritmo para seleccionar el segmento más adecuado con el fin de crear una reproducción fluida. Hay varias especificaciones de formato HAS, pero las más utilizadas son *HTTP Live Streaming* (HLS) y *Dynamic Adaptive Streaming over HTTP* (DASH). Estos sistemas de transmisión requieren sistemas de codificación de vídeo eficientes para entregar el contenido en tantas resoluciones como sea necesario. Además, para ofrecer la mejor QoE posible (Seufert et al., 2015), estos vídeos deben alcanzar la calidad más alta en la fase de codificación, a fin de reducir las probables causas de degradación de la QoE.

Uno de los parámetros de codificación más comunes disponibles para la transmisión en vivo y bajo demanda es el control del *bitrate* o tasa de bits; que se ha utilizado desde MPEG-1 (ISO Central Secretary, 1993), y dicta cómo el codificador asigna cuántos bits se utilizarán para cada estructura de grupo de imágenes (Group of Pictures -GOP-) que impone imágenes Intra periódicas (que es un frame que se puede decodificar sin información de otros frames). Esto determinará el tamaño del archivo y también cómo se distribuye la calidad —valores altos de *bitrate* se corresponden con una mayor calidad, pero también con archivos de mayor tamaño—. Actualmente, existen diferentes métodos disponibles para controlar el *bitrate*, entre los más utilizados están i)

Constant Bitrate (CBR) - se refiere a la codificación/transcodificación de una transmisión con un valor relativamente fijo de *bitrate* y ii) *Variable Bitrate* (VBR) - se refiere a la codificación/transcodificación de segmentos de vídeo con diferentes valores de *bitrate*, dependiendo de la complejidad del contenido.

Por consiguiente, si el *bitrate* dicta la cantidad de datos transferidos durante un tiempo determinado, entonces un *bitrate* constante, CBR, es un flujo aproximadamente constante de datos por unidad de tiempo. Esto lo hace factible, en el sentido que, se puede establecer una tasa de bits fija y saber con exactitud lo que se va ha transmitir. Sin embargo, esta técnica no se adapta al tamaño ni a la complejidad de los datos transmitidos porque se asigna el mismo número de bits tanto a segmentos simples (es decir, segmentos con escenas de bajo movimiento espacio temporal —SI & TI–[1] o baja complejidad) como a los segmentos complejos (es decir, segmentos con escenas de alto movimiento espacio temporal o alta complejidad), provocando una variación de calidad entre segmentos y posiblemente un efecto negativo en la satisfacción del usuario. Por consiguiente, este enfoque no optimizado podría conducir a una calidad general más baja, según la tasa de bits objetivo y los objetivos de transmisión. Este método es particularmente útil en escenarios de transmisión en directo o *Live-to-VOD*; donde la regla general es elegir una tasa de bits alta para garantizar una transmisión de alta calidad.

El VBR, por el contrario, se refiere a una transmisión donde la tasa de bits es variable en el tiempo, ya que codifica segmentos simples con menos bits y segmentos complejos con más bits; mientras mantiene un nivel constante de calidad a lo largo de toda la secuencia de vídeo. La capacidad de obtener una mejor calidad de vídeo para la misma tasa de bits promedio, o la misma calidad con una codificación de tasa de bits más baja que el CBR son algunas de las ventajas clave que tiene el VBR sobre el CBR (Lakshman et al., 1998).

Históricamente, las plataformas de transmisión solo usaban CBR, en parte porque no hacerlo implicaba una serie de desafíos prácticos, como la compleja implementación para la codificación de VBR y los desafíos que supone el almacenamiento, la recuperación y el transporte que plantea las múltiples escalas de *bitrate* de los vídeos VBR (Qin et al., 2018). Con las mejoras sustanciales en la relación *calidad-bitrate* frente a CBR y los avances tecnológicos en las prácticas de codificación de VBR, esto ha llevado a que los

[1]SI & TI: Como se describe en la Recomendación (ITU-T, 2022), la información espacial (SI) y la información temporal (TI) son métricas ampliamente utilizadas como una aproximación de la complejidad de la escena y se usan con frecuencia para la selección de secuencias de prueba y las tareas relacionadas con la evaluación de la calidad.

proveedores de contenido y la investigación académica comience a explorar y utilizar VBR (Huang et al., 2014; Reed y Klimkowski, 2016; Qin et al., 2018; Huang et al., 2021).

Con esta premisa, si queremos optimizar el proceso de codificación VBR para cada resolución y segmento, ¿qué tasa de bits o CRF (Constant Rate Factor) debemos seleccionar en cada momento? ¿Es esta selección independiente del contenido/tipo de vídeo? ¿Cuál es la mayor tasa de bits o CRF necesaria para conseguir la mejor calidad perceptible?.

Para responder a estas preguntas, presentamos dos sistemas de codificación VBR capaces de codificar un vídeo a partir de un valor de métrica objetiva de calidad, como VMAF (Video Multi-method Assessment Fusion) (Li et al., 2016b).

La propuesta a nivel general sería la siguiente: partiendo de un vídeo en en formato crudo o de muy alta calidad en resolución 4K y de una calidad objetivo, los sistemas propuestos utilizan una versión espacialmente reducida del vídeo para seleccionar el valor de CRF con el que codificar cada segmento para que se ajuste al valor objetivo de calidad. Utilizando estos valores de este parámetro de codificación, los sistemas codifican cada segmento de forma independiente a las resoluciones seleccionadas. Finalmente, los segmentos se concatenan o "ensamblan" para producir una versión codificada de la secuencia de vídeo completa, para luego utilizarlas en una transmisión de VOD sobre DASH.

Tanto la primera[2] como la segunda[3] aproximación han sido evaluadas y comparadas con una codificación basada en un valor fijo de CRF. Los resultados de ambas propuestas muestran mejoras de más del 10 % en cuanto al tamaño final del vídeo, manteniendo el valor de calidad indicado por el usuario.

[2]Una comparación visual entre la primera propuesta y una codificación con CRF fijo puede verse en: https://links.uv.es/jgutierr/qinloopcoding

[3]La comparación visual para la segunda propuesta en: https://links.uv.es/jgutierr/multiresQ

3.2. Esquema I: Propuesta de codificación de vídeo basado en la calidad

Esta sección describe el proceso utilizado para la preparación de las representaciones de vídeo para DASH, teniendo en cuenta la segmentación para este tipo de transmisión de contenido.

A diferencia de otras propuestas expuestas en la subsección 2.5.1, nuestra propuesta incorpora la calidad de vídeo, utilizando la métrica VMAF, como una entrada al sistema de codificación, con el objetivo de seleccionar los parámetros de codificación adecuados para cada segmento de vídeo. Cada representación puede ser caracterizada por un valor de calidad objetivo preestablecido por el usuario, que corresponde a un valor de calidad constante en todo el vídeo. Además, estas secuencias se codifican para una gama finita de resoluciones manteniendo el mismo nivel de calidad mediante un solo proceso de codificación —una pasada—.

3.2.1. Algoritmo de codificación basada en calidad

En esta subsección se describe el algoritmo desarrollado y cada uno de los componentes que conforman el sistema de codificación. En ella también se explica brevemente el proceso de análisis de calidad y despliegue. El sistema de codificación se ha diseñado en dos componentes con el fin de ampliar su usabilidad y su despliegue en sistemas de procesamiento distribuido, tal y como se muestra en el diagrama de bloques (ver Figura. 3.1).

El primer componente, denominado **Core-Loop**, se encarga de procesar el contenido y obtener la información necesaria para codificar el vídeo a una calidad objetivo especificada por el usuario. Este componente toma como entrada el vídeo en crudo en su resolución original, 4K en nuestro caso. El vídeo en crudo, v, es procesado por el ***Módulo de Dimensionalidad***, que produce un vídeo en crudo v' a una resolución menor (por ejemplo, 240p) manteniendo la relación de aspecto del vídeo original. Después de este proceso, v' es utilizado por el ***Módulo de Segmentación*** para dividir el vídeo en segmentos de duración fija $\{v'_1, \cdots, v'_N\}$. Una vez segmentado el vídeo, en el ***Módulo de Preparación de Datos*** se codifica cada segmento v'_i a diferentes valores de CRF $\{10, 16, 22, 28, 34, 40, 46, 51\}$. Todos estos segmentos codificados se pasan al ***Módulo de Calidad*** junto con el segmento origi-

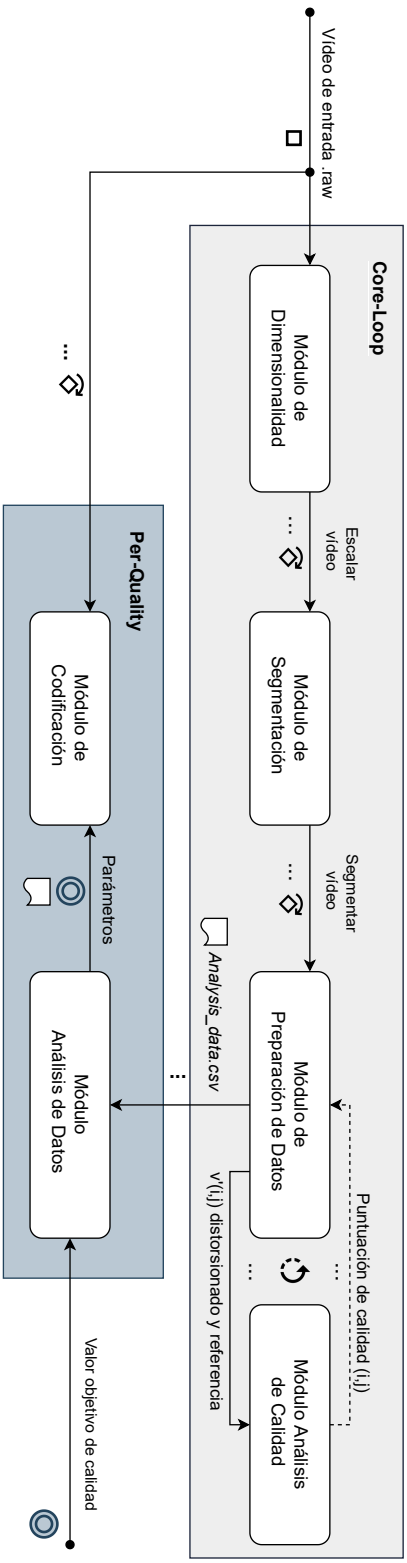
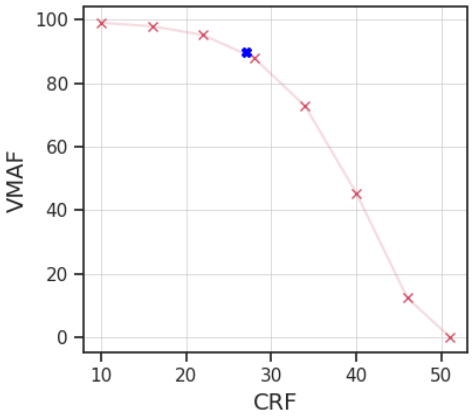
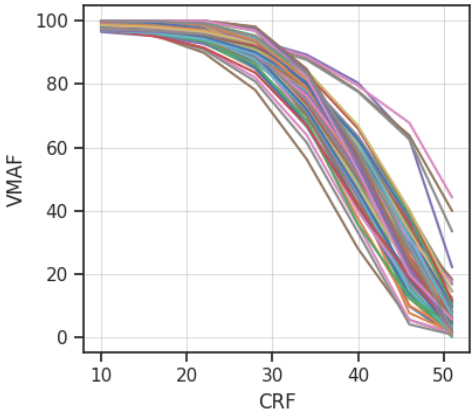


Figura 3.1: Esquema I: Propuesta de codificación DASH basada en la calidad.



(a) Puntos muestreados y valor de CRF recomendado para un segmento.



(b) Curvas para la interpolación.

Figura 3.2: Curvas utilizadas para determinar el valor del CRF ajustado a la complejidad de cada escena dado un valor objetivo de calidad.

nal para obtener el valor de la calidad (en este caso VMAF). Esto significa que para cada segmento se genera un conjunto de puntos $\{(CRF_j, Q_j)\}$. La Figura. 3.2a muestra los puntos obtenidos para un segmento junto al valor recomendado de CRF para un valor objetivo VMAF de 90 y la Figura. 3.2b todas las curvas obtenidas para todos los segmentos del vídeo Sintel[4].

El segundo componente, llamado **Per-Quality**, toma los datos generados y la calidad objetivo Q_t. Esta información se pasa al ***Módulo de Análisis de Datos*** cuyo objetivo es encontrar el valor de CRF para cada segmento con el que se obtiene la calidad objetivo. Para cada segmento es posible definir la función $Q(CRF)$ interpolando los puntos $\{(CRF_j, Q_j)\}$, como se muestra en la Figura. 3.2a. A partir de esta interpolación es posible encontrar el valor de CRF óptimo a aplicar al segmento para conseguir la calidad objetivo Q_t.

Una vez calculados los valores CRF, cada segmento puede ser codificado para diferentes resoluciones manteniendo la relación de aspecto del vídeo original. Para este proceso, el ***Módulo de Codificación*** utiliza como entrada el vídeo original v, el valor CRF calculado para cada segmento y el rango de resoluciones.

La solución propuesta prepara el contenido de vídeo para las transmisiones DASH bajo demanda con la calidad deseada. La calidad de las diferentes representaciones que componen la secuencia dependerá únicamente del valor

[4]https://mango.blender.org

de calidad objetivo definido por el usuario.

3.2.2. Experimentos y Evaluación del Rendimiento

3.2.2.1. Información del Entorno de Pruebas

En este trabajo, los vídeos se han codificado utilizando x264[5] y como métrica de calidad se ha usado VMAF[6] en su versión v2.0.0. Sin embargo, la solución propuesta puede utilizar diferentes codificadores y diversas métricas. Para evaluar y validar el método propuesto, se han realizado varias pruebas sobre dos vídeos: Tears of Steel (TOS)[7] y Sintel (STL)[8].

Estos vídeos tienen una resolución espacial de 4K con una tasa de fotogramas de 24 fps, y están almacenados en formato YUV4Mpeg (y4m), tal y como se resume en la Tabla 3.1. Además, para aproximar nuestra propuesta a un entorno real, la duración de los vídeos es superior a 10 minutos.

Tabla 3.1: Características de las secuencias de vídeo utilizadas para la evaluación y la validación.

Vídeo	**Resolución**	**fps**	**Fotogramas**	**Tamaño (GB)**	**Categoría**
STL	4096x1744	24	21312	213	Fantasía
TOS	4096x1714	24	17620	173	Acción

El hardware/software utilizado en todos los experimentos se especifica en la Tabla 3.2. Se han utilizado contenedores Docker[9] con el objetivo de aprovechar la capa de abstracción y automatización en la virtualización de aplicaciones, y su capacidad de ser desplegados en múltiples sistemas operativos e infraestructuras. Se utilizan dos contenedores, el primero implementa el sistema de codificación con sus diversos componentes para la preparación del contenido en múltiples representaciones. El segundo contenedor se utiliza para el análisis de calidad y ofrece soporte para diferentes métricas, tal y como se detallará en el capítulo 6 de esta tesis. El contenedor desarrollado

[5]ffmpeg con la biblioteca del códec libx264
[6]https://github.com/Netflix/vmaf/releases/tag/v2.0.0
[7]https://durian.blender.org
[8]https://mango.blender.org
[9]https://www.docker.com/

puede tomar como entrada el vídeo distorsionado y el vídeo de referencia para el análisis de las métricas con referencia completa y referencia reducida; o sólo el vídeo codificado para las métricas sin referencia.

Tabla 3.2: Características del hardware utilizado en las pruebas

Atributo	**Detalles**
CPU	Intel(R) Core(TM) i7-4790 @3.60GHz
Cache size	8192 KB
Placa Base	ASUS MB H81M-D PLUS
Memoria RAM	Kingston 99U5403-159.A01LF 2x8 GB DIMM DDR3 1600 MHz
Almacenamiento	WD40NDZW-11A8JS1, 5400 RPM
SO	Linux Mint 17.3 Rosa x64 bits
Docker Versión	19.03.5

3.2.2.2. Experimentos y Resultados

En esta subsección presentamos los experimentos realizados y los resultados obtenidos utilizando el esquema de codificación propuesto. Como hemos indicado en el apartado anterior, disponemos de 2 vídeos en formato Y4M (STL y TOS) con resolución 4K. Cada vídeo ha sido codificado para siete tamaños de segmento (1, 2, 3, 5, 8, 10 y 12 segundos) y para siete resoluciones (240p, 360p, 480p, 720p, 1080p, 1440p y 2160p). Sin embargo, en la presentación de este primer esquema nos centraremos en la resolución 1080p, pero hay que señalar que los resultados mostrados con esta resolución son similares para las demás resoluciones evaluadas. Nuestra propuesta se ha comparado con una codificación basada en segmentos utilizando un CRF fijo.

a) Selección de valores constantes del CRF

Para obtener el valor constante de CRF con el que comparar nuestra propuesta, hemos codificado los vídeos completos utilizando diferentes valores de CRF como se muestra en la Tabla 3.3. Con un CRF de 27 el VMAF obtenido es mayor o igual a 90, que es el valor de calidad objetivo utilizado como entrada en nuestro sistema.

b) Experimentos con diferentes tamaños de segmentos

La distribución de los CRF para los diferentes tamaños de segmento de

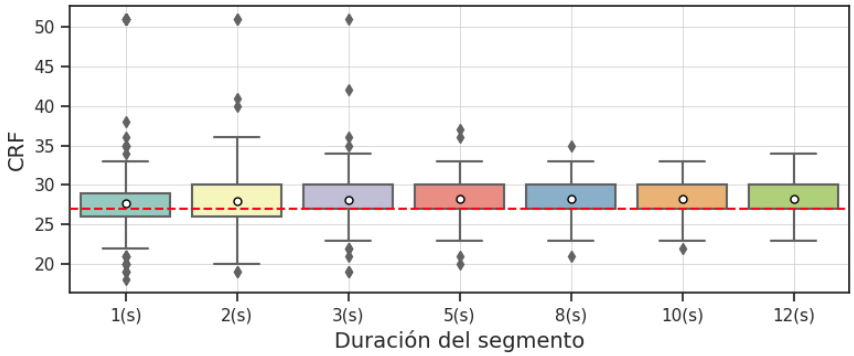
Tabla 3.3: Vídeos codificados con diferentes CRF (sin segmentar)

CRF	STL			TOS		
	VMAF	***Tiempo (min)***	***Tamaño (MB)***	***VMAF***	***Tiempo (min)***	***Tamaño (MB)***
15	98.504	21.035	1554.066	98.810	20.230	2676.059
21	97.057	20.724	642.715	97.005	17.669	724.297
23	96.137	19.145	484.047	95.912	15.951	506.457
25	94.851	18.867	366.625	94.415	15.816	370.617
27	92.151	18.743	281.148	92.376	15.600	280.242
29	90.687	18.685	218.363	89.694	15.534	216.645
31	87.552	18.591	171.367	86.225	15.366	170.145
35	78.657	18.371	109.234	76.791	15.332	108.871

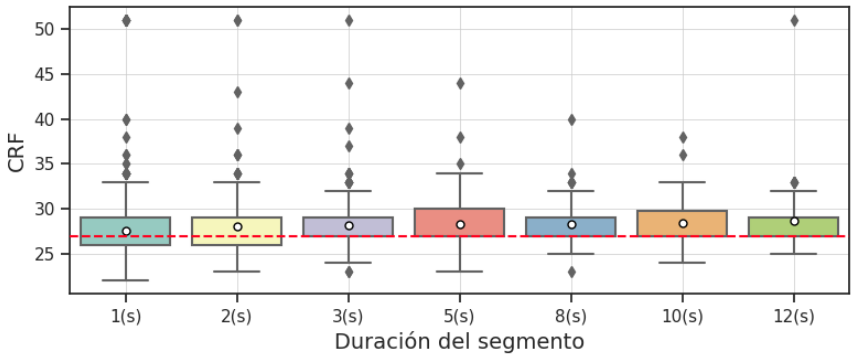
los dos vídeos considerados se muestran en la Figura. 3.3. Como se puede observar, la media es muy cercana a 27, pero el esquema propuesto permite la adaptación al contenido del vídeo. Cabe destacar que estos valores se han obtenido para la dimensión reducida (240p) y se han aplicado al resto de las siete resoluciones indicadas al inicio de esta subsección.

Las Tablas 3.4 y 3.5 presentan un resumen de los datos obtenidos durante la codificación de diferentes tamaños de segmento con los vídeos STL y TOS, respectivamente. Las tablas muestran, para cada duración de segmento, el número de segmentos, la media aritmética de VMAF, la desviación estándar de VMAF, el tiempo de codificación y el tamaño del vídeo codificado. Estos datos se muestran para una codificación de CRF fijo de 27 y para CRF adaptativo (nuestra propuesta) con un VMAF objetivo de 90. Finalmente, la última columna presenta una comparación entre el tamaño de la codificación CRF fija y adaptable. Un valor negativo significa que la codificación fija mejora nuestra propuesta mientras que un valor positivo significa que nuestra propuesta aporta beneficios.

En la película Sintel (ver Tabla 3.4), podemos ver que nuestra propuesta tiene un VMAF medio de 92.17, mientras que el VMAF medio de la codificación CRF fija es de 92.79. Esta diferencia de 0.62 en la métrica VMAF no es perceptible para los humanos y menos aún cuando se trata de valores VMAF superiores a 88 (considerados de muy alta calidad). Nuestra propuesta tiene una desviación estándar menor, lo que implica un mejor ajuste a la calidad objetivo.



(a) Vídeo STL



(b) Vídeo TOS.

Figura 3.3: Distribución de los CRFs calculados para diferentes tamaños de segmento.

Si nos fijamos en el tamaño del vídeo codificado, nuestra propuesta mejora en todos los casos. Tenemos un porcentaje de mejora superior al 8.4 % cuando los segmentos tienen un tamaño igual o superior a 3 segundos. En términos medios, nuestra propuesta mejora el tamaño del vídeo en un 8.8 %.

En el vídeo Tears of Steel (véase la Tabla 3.5), podemos ver que nuestra propuesta tiene un VMAF medio de 91.31, mientras que el VMAF medio de la codificación CRF fija es de 92.07. Esta diferencia de 0.76, inferior a 1 y similar a la obtenida con el vídeo STL. Nuestra propuesta tiene una desviación estándar menor, lo que implica un mejor ajuste a la calidad objetivo, en este caso en torno a 90.

Si nos fijamos en el tamaño del vídeo codificado, nuestra propuesta mejora en todos los casos, excepto con segmentos de 1 segundo. Tenemos un

porcentaje de mejora superior al 6.78 % cuando los segmentos tienen un tamaño igual o superior a 3 segundos. En términos medios, nuestra propuesta mejora el tamaño del vídeo en un 7.5 %.

Los tiempos de codificación con CRF fijo y CRF adaptativo son similares. En el caso adaptativo el tiempo mostrado corresponde al tiempo empleado para codificar el video a la resolución de 1080p. El tiempo medio empleado por el Core Loop (que calcula los valores de CRF para cada segmento a la resolución de 240p) es 12,28 min para STL y 16,36 min para TOS. No se suma este tiempo a las Tabla 3.4 y 3.5 puesto que esta etapa se realiza una sola vez para todo el rango de resoluciones seleccionadas; en nuestro caso las siete resoluciones mencionadas al inicio de está sección.

c) Análisis de calidad para segmentos de 5s

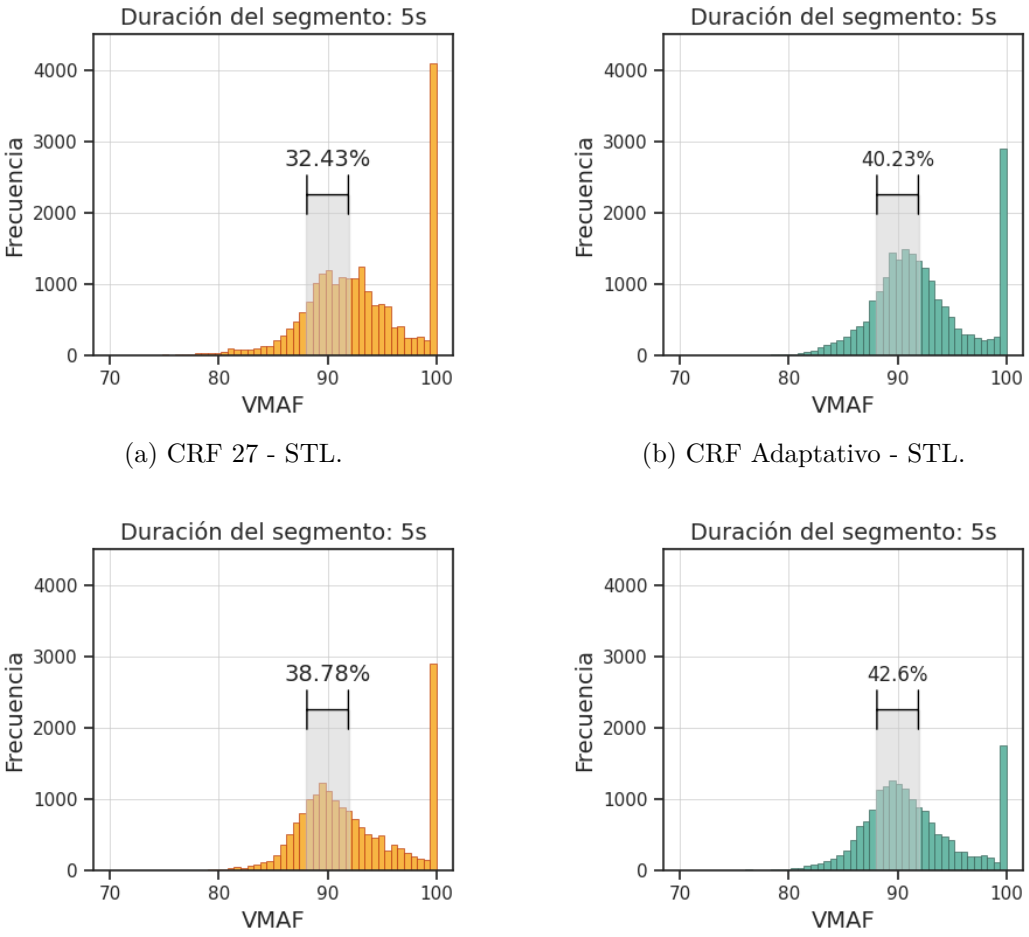
Varias empresas dedicadas a laS transmisiones HAS indican que el tamaño de los segmentos puede variar mucho, ya que la calidad percibida por el usuario depende de éste y otros muchos factores. En (Seufert et al., 2015), los autores recomiendan un tamaño de segmento igual o inferior a 6 segundos, por lo que centraremos nuestro estudio en la codificación con segmentos de 5 segundos.

Tabla 3.4: Comparación de los datos del proceso de codificación para Sintel para la resolución 1080p.

STL		**CRF Fijo**				**CRF Adaptativo**				**Fijo vs. Adaptativo**
Duración Segmentos	***Núm. Segmentos***	***VMAF***	***Std dev***	***Tiempo (min)***	***Tamaño (MB)***	***VMAF***	***Std dev***	***Tiempo (min)***	***Tamaño (MB)***	***Tamaño (%)***
1(s)	888	92.269	5.085	38.481	321.062	92.326	4.226	34.839	315.590	1.704
2(s)	444	92.657	4.958	36.863	297.793	92.240	4.355	33.238	278.422	6.505
3(s)	296	92.843	4.952	33.163	291.078	92.176	4.479	27.453	266.523	8.436
5(s)	178	92.896	4.857	24.457	284.039	92.158	4.486	25.206	255.891	9.910
8(s)	111	92.896	4.857	23.304	280.727	92.139	4.531	21.760	250.270	10.849
10(s)	89	92.977	4.825	22.532	279.672	92.103	4.629	21.961	247.285	11.580
12(s)	74	92.988	4.818	21.049	282.645	92.108	4.662	23.141	250.184	11.485

Tabla 3.5: Comparación de datos del proceso de codificación para Tears of Steel para la resolución 1080p.

TOS		**CRF Fijo**				**CRF Adaptativo**				**Fijo vs. Adaptativo**
Duración Segmentos	***Núm. Segmentos***	***VMAF***	***Std dev***	***Tiempo (min)***	***Tamaño (MB)***	***VMAF***	***Std dev***	***Tiempo (min)***	***Tamaño (MB)***	***Tamaño (%)***
1(s)	735	91.457	4.936	31.330	308.406	91.513	4.193	28.772	315.964	-2.451
2(s)	368	91.905	4.798	26.026	292.754	91.363	4.360	26.159	279.816	4.419
3(s)	245	92.052	4.744	25.964	287.355	91.280	4.347	25.493	267.864	6.783
5(s)	147	92.212	4.703	21.815	283.430	91.260	4.507	23.576	256.788	9.400
8(s)	92	92.294	4.666	18.483	281.633	91.313	4.445	18.628	252.936	10.190
10(s)	74	92.304	4.652	19.345	280.676	91.189	4.600	18.961	248.240	11.556
12(s)	62	92.298	4.650	17.983	282.074	91.263	4.587	18.606	248.996	11.727



(a) CRF 27 - STL.

(b) CRF Adaptativo - STL.

(c) CRF 27 - TOS.

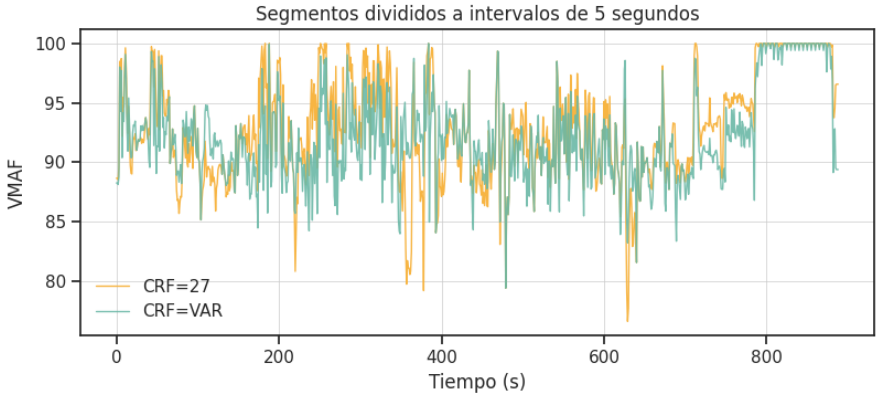
(d) CRF Adaptativo - TOS.

Figura 3.4: Distribución VMAF para dos vídeos, STL superior y TOS inferior, con un segmento de 5s.

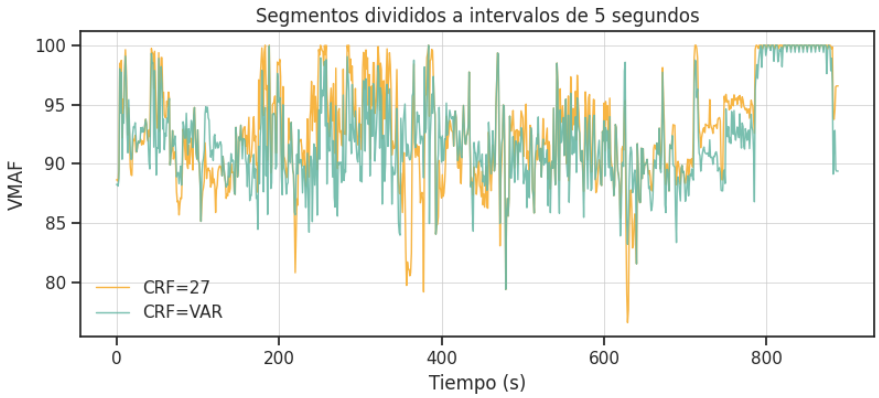
Para comprobar si nuestro sistema se ajusta a la calidad objetivo (en este caso VMAF=90) cuando los dos vídeos están codificados a una resolución de 1080p, hemos obtenido la calidad VMAF de cada fotograma. La Figura. 3.4 muestra la distribución de VMAF para los dos vídeos STL (arriba) y TOS (abajo) y los dos esquemas de codificación: CRF constante (izquierda) y CRF adaptativo (derecha) para un tamaño de segmento de 5s.

La Figura. 3.4 muestra un porcentaje en cada figura. Este porcentaje corresponde al número de fotogramas que tienen un VMAF entre 88 y 92. En STL vemos que nuestra propuesta tiene un 40.23 % de los fotogramas en torno a la calidad objetivo (90), aproximadamente un 7 % más que la solución

con CRF fijo. En TOS, nuestra propuesta tiene un 42.6 % de los fotogramas alrededor de la calidad objetivo, ligeramente superior al 38.78 % obtenido con CRF fijo. Sin embargo, vemos que hay más valores de VMAF muy cercanos a 100 en el caso de CRF constante. La Figura 3.5 muestra la distribución de VMAF por fotograma para los dos vídeos y esquemas de codificación. Podemos ver que el caso de CRF constante da valores muy cercanos a 100 en los fotogramas finales (créditos finales) en ambos vídeos mientras que nuestra propuesta da valores de calidad ligeramente inferiores.



(a) STL



(b) TOS

Figura 3.5: VMAF por fotograma para los dos vídeos.

3.3. Esquema II: Propuesta de codificación de vídeo basado en la calidad con un algoritmo iterativo

En esta sección se describe otra propuesta de esquema de codificación y el algoritmo utilizado para obtener adaptativamente el CRF de cada segmento. El funcionamiento del sistema propuesto y su esquema de codificación se muestra en la Figura 3.6.

3.3.1. Algoritmo de codificación basada en calidad

El usuario proporciona tres entradas al sistema:

1. un vídeo de alta calidad o crudo,
2. una calidad objetivo, y
3. una lista de resoluciones.

El vídeo de entrada, v, se da a la máxima resolución, por ejemplo, en resolución 4K, y luego v se reduce a una resolución inferior v'. Esta reducción de la dimensionalidad se realiza para acelerar el proceso de análisis. El vídeo de baja dimensión se divide en segmentos de duración uniforme. A continuación, cada segmento v'_i se procesa para obtener el CRF que debe aplicarse en el codificador para alcanzar la calidad objetivo proporcionada por el usuario. Por último, el ***Módulo de Codificación Adaptativa*** de segmentos codifica los segmentos del vídeo original en bruto a diferentes resoluciones utilizando los valores de CRF calculados en el ***Módulo de Análisis de Segmentos***.

El ***Módulo de Análisis de Segmentos*** codifica primero el segmento utilizando el CRF inicial, y luego se calcula la calidad del vídeo codificado en el ***Módulo de Calidad***. Este proceso se repite, adaptando el CRF, hasta alcanzar la calidad objetivo Q_t. Este procedimiento equivale a resolver el problema convexo

$$\arg\min_{\text{CRF}} |Q - Q_t|$$

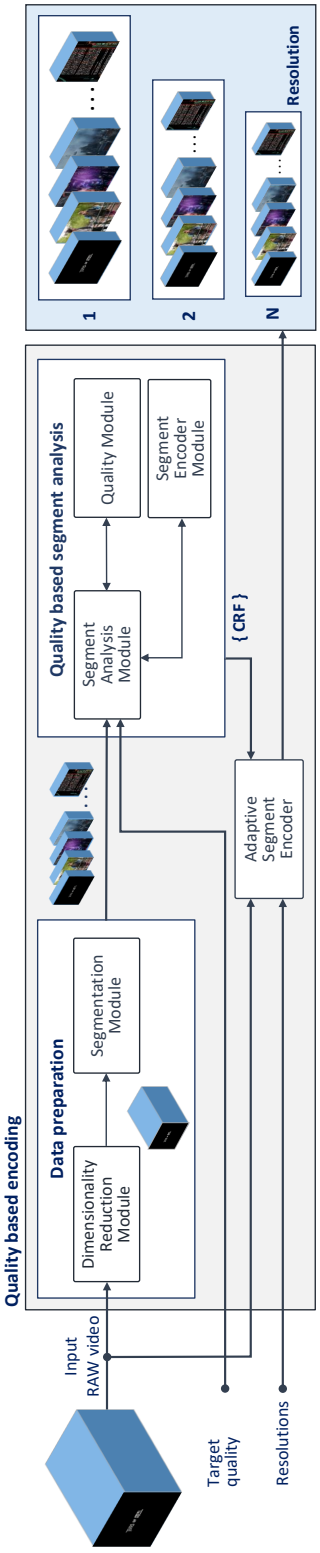


Figura 3.6: Esquema II: Propuesta de codificación DASH basada en la calidad.

donde la función Q, que depende del CRF, es desconocida. Por lo tanto, este es un problema de optimización *online* y se ha resuelto utilizando el método del descenso de gradiente con un paso adaptativo como se muestra en el Algoritmo 1.

Algorithm 1: Estimación iterativa de CRF usando el vídeo con baja resolución.

Input: Raw Video Segment v_i'
Input: Target quality Q_t
Input: Initial CRF
Output: CRF to achieve Q_t in segment v_i'

$Threshold \leftarrow 1$
v_i'' =Segment_Encoder_Module(v_i',CRF)
Q=Quality_Module(v_i', v_i'')
$step \leftarrow \text{sign}(Q - Q_t) * 6$
$changed \leftarrow False$
while $\big((|Q - Q_t| \geqslant Threshold)\,\&\,(|step| > 0)\big)$ **do**
  $CRF \leftarrow CRF + step$
  v_i'' =Segment_Encoder_Module(v_i',CRF)
  Q=Quality_Module(v_i', v_i'')
  $stepn \leftarrow \text{sign}(Q - Qt) * |step|$
  if $stepn * step < 0$ **then**
    $changed \leftarrow True$
  end
  if $changed$ **then**
    $stepn \leftarrow \lfloor \frac{2 \cdot stepn}{3} \rfloor$
  end
  $step \leftarrow stepn$
end
return CRF

El ***Módulo de Análisis de Segmentos*** recibe el segmento de vídeo escalado (v_i'), el CRF inicial y la calidad objetivo. El primer paso es codificar el segmento de vídeo y obtener su calidad (los comandos `ffmpeg` se ven en la Listing 11).

Si la calidad calculada es mayor que la calidad objetivo, el CRF debe aumentarse; en caso contrario, el CRF debe reducirse. Este proceso se itera hasta la convergencia. El paso inicial se mantiene hasta que su signo debe ser cambiado, a continuación, en cada iteración su valor absoluto se reduce

```
# Codificar segmento reducido
ffmpeg -y -i segment_p.y4m -c:v libx264 -crf CRF segment_encoded.mp4

# Obtener calidad (usando ffmpeg)
ffmpeg -i segment_encoded.mp4 -i segment_p.y4m -lavfi libvmaf -f null -
↪  2>&1
```

Listing 1: Ejemplo simple para codificar y obtener la calidad de un segmento reducido.

en un factor de $\frac{2}{3}$. Para reducir el número de iteraciones realizadas en el bucle de optimización mostrado en el Algoritmo 1, como es probable que los segmentos adyacentes sean similares, el CRF_i obtenido para el segmento i se utiliza como valor inicial para el análisis del siguiente segmento $i + 1$.

La salida del ***Módulo de Análisis de Segmentos*** es una secuencia de valores CRF, $\{\text{CRF}_i\}$ que se utilizan para codificar los segmentos del vídeo original de resolución completa a diferentes resoluciones.

3.3.2. Experimentos y Evaluación del Rendimiento

Esta sección muestra los experimentos realizados para validar y evaluar el sistema propuesto. En la subsección 3.3.2.1 se presenta el banco de pruebas utilizado para realizar los experimentos, en la subsección 3.3.2.2 se analiza un caso base para encontrar una referencia con la que comparar nuestra propuesta, y en el resto de subsecciones se presentan y analizan los datos obtenidos en las pruebas.

3.3.2.1. Información del Entorno de Pruebas

En nuestra evaluación de rendimiento se han realizado varias pruebas sobre tres películas de código abierto: Big Buck Bunny (BBB) [10], Sintel (STL)[11] y Tears of Steel (TOS)[12]. Estos vídeos tienen una resolución espacial de 4K con una tasa de fotogramas de 24 fps, y están almacenados en formato YUV4Mpeg (y4m). Además, para aproximar la propuesta a un entorno real,

[10]https://peach.blender.org/
[11]https://mango.blender.org
[12]https://durian.blender.org

Tabla 3.6: Secuencias de Vídeo para Evaluación y Validación.

Vídeo	Resolución	fps	Fotogramas	Tamaño (GB)	Categoría
BBB	3840x2160	24	15231	177	Dibujos animados
STL	4096x1744	24	21312	213	Fantasía
TOS	4096x1714	24	17620	173	Acción

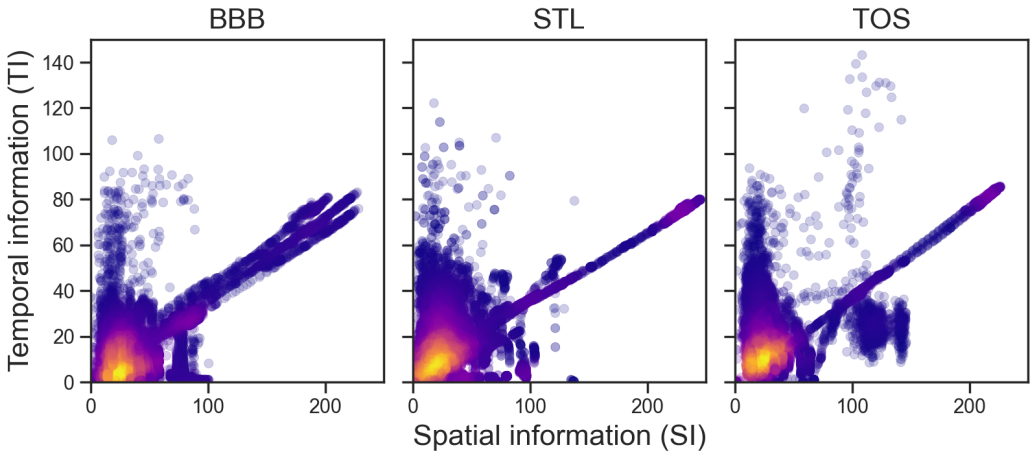


Figura 3.7: Mapa de calor de la información Espacio-Temporal (SI-TI) de todos los fotogramas.

la duración de los vídeos es superior a 10 minutos. La información de las secuencias de vídeo se presenta en la Tabla 3.6, y su complejidad espacial y temporal se muestra en la Figura 3.7.

En este trabajo, estos vídeos se han codificado utilizando x264[13] y se ha utilizado VMAF como métrica de calidad objetivo. Sin embargo, la solución propuesta podría utilizar diferentes codificadores y varias métricas. En nuestros experimentos, hemos seleccionado un tamaño de segmento de 4 segundos con el fin de obtener un buen compromiso entre la eficiencia de la codificación y la flexibilidad para la adaptación del flujo a los cambios de ancho de banda (Lederer, 2015).

El hardware/software utilizado en todos los experimentos se especifica en la Tabla 3.2. El esquema de codificación mostrado en la Figura 3.6 se ha encapsulado en dos contenedores Docker. Uno de los contenedores encapsula el sistema de codificación con sus diversos componentes para la preparación

[13]Herramienta FFmpeg con la biblioteca de códecs libx264

de contenidos en múltiples representaciones y configuraciones. El segundo contenedor encapsula el análisis de calidad y ofrece soporte para diferentes métricas. Los dos contenedores comparten un volumen local para minimizar las operaciones de lectura/escritura en disco o el tráfico de red.

3.3.2.2. Caso base: codificación del vídeo completo

Con el fin de comparar el esquema adaptativo propuesto con una codificación CRF de fijo por segmento, hemos codificado los tres vídeos —completo sin procesar—, a una resolución de 1080p, variando el valor CRF de 25 a 31. La Tabla 3.7 muestra por cada vídeo VMAF, SSIM, PSNR, tiempo de codificación y su tamaño para cada CRF utilizado. Comparando los resultados de esta Tabla con los obtenidos por el enfoque propuesto mostrado en la Tabla 3.8 para 1080p, un valor de CRF de 27 da un valor similar de VMAF. Por lo tanto, este es el valor seleccionado para codificar todos los segmentos en la codificación CRF fija.

Tabla 3.7: Vídeos codificados con diferentes CRF (sin segmentar) a resolución 1080

Vídeo	**CRF**	**VMAF**	**SSIM**	**PSNR**	**Tiempo (min)**	**Tamaño (MB)**
BBB	25	95.112	0.997	43.693	25.783	174.160
	27	93.575	0.996	42.486	25.798	136.129
	29	91.524	0.995	41.271	25.782	107.086
	31	88.886	0.993	40.059	25.780	84.699
STL	25	94.851	0.996	43.669	18.867	366.625
	27	92.151	0.995	42.648	18.743	281.148
	29	90.687	0.993	41.600	18.685	218.363
	31	87.552	0.991	40.538	18.591	171.367
TOS	25	94.415	0.995	41.327	15.816	370.617
	27	92.376	0.993	40.482	15.600	280.242
	29	89.694	0.991	39.588	15.534	216.645
	31	86.225	0.988	38.649	15.366	170.145

3.3.2.3. Iteraciones y adaptación CRF

En esta subsección vamos a analizar cómo funciona nuestro algoritmo iterativo de estimación CRF y cómo afecta al rendimiento del sistema de codificación propuesto. El primer valor a analizar es el tiempo medio necesario para obtener el valor CRF que proporciona la calidad objetivo. Este tiempo cambia para cada vídeo probado, en BBB obtenemos un tiempo medio de 1.01 segundos por segmento. En STL este valor medio aumenta a 1.53 segundos y en TOS ronda los 1.61 segundos. Si atendemos a las iteraciones medias por segmento, en BBB tenemos 2.60 interacciones para encontrar un CRF adecuado y este valor aumenta hasta 2.94 para STL y 2.92 para TOS.

El tiempo medio en todas las secuencias de vídeos se sitúa en torno a 1.38 segundos. Obteniendo un coste medio por escena del 34 % en términos de tiempo y mejorando en un 10 % el coste requerido por el sistema propuesto en (Xing et al., 2019).

Como hemos visto en la sección anterior, nuestro sistema ajusta el CRF para obtener la calidad objetivo. La Figura 3.8 muestra la distribución de frecuencias de los valores CRF utilizados para codificar los segmentos de cada vídeo. La distribución para el vídeo BBB presenta menos variabilidad, oscilando entre 24 y 34. Los CRF más seleccionados son 28, 27 y 30 según la frecuencia de uso. En el caso de STL y TOS hay mayor variabilidad. Por ejemplo, en STL el rango de CRFs seleccionados va de 20 a 43, siendo los valores de CRF más comunes: 28, 27 y 26. Algo similar ocurre con TOS, pero en este caso el CRF más seleccionado proporcionado por nuestro sistema es 27 seguido de 26.

3.3.2.4. Comparación adaptativa frente a fija

La Figura 3.9 representa la diferencia de tamaño de cada segmento de vídeo codificado con un CRF adaptativo y un CRF fijo, donde valores negativos representan una disminución (mejora) de tamaño en la codificación adaptativa debido a la selección de un valor de CRF superior a 27. Los valores positivos implican un mayor tamaño en el segmento codificado utilizando el algoritmo adaptativo debido a la selección de un valor de CRF inferior a 27. En todos los casos, el algoritmo adaptativo selecciona el valor CRF que alcanza la calidad VMAF objetivo para cada segmento.

La Figura 3.10 muestra el SI normalizado, el TI y la diferencia de tamaño

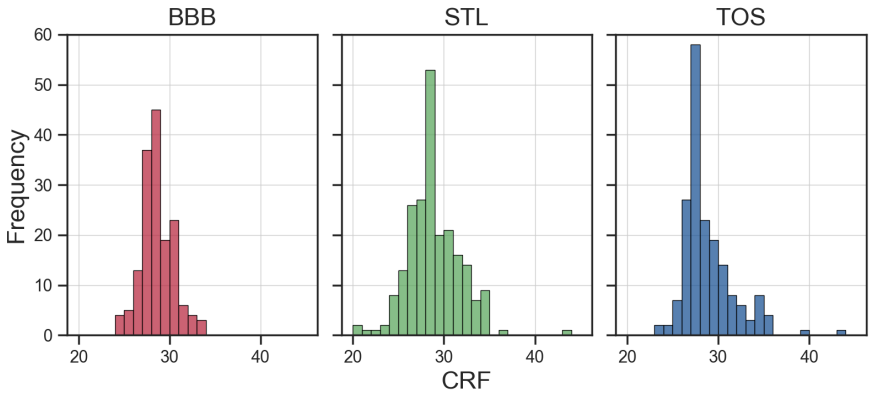


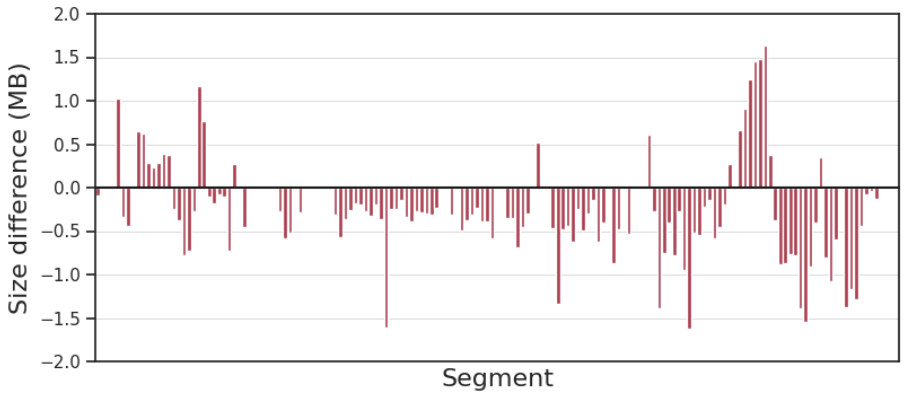
Figura 3.8: Distribución CRF de los valores obtenidos por el algoritmo adaptativo para cada vídeo.

para STL a resolución 4K. Como puede observarse, se obtiene una mayor ganancia de tamaño utilizando el enfoque adaptativo para aquellos segmentos que tienen un valor grande de TI. Algo similar ocurre con otras secuencias. De acuerdo con esta información, podemos decir que nuestro sistema tiene una mayor capacidad de compresión a medida que aumenta la complejidad del vídeo.

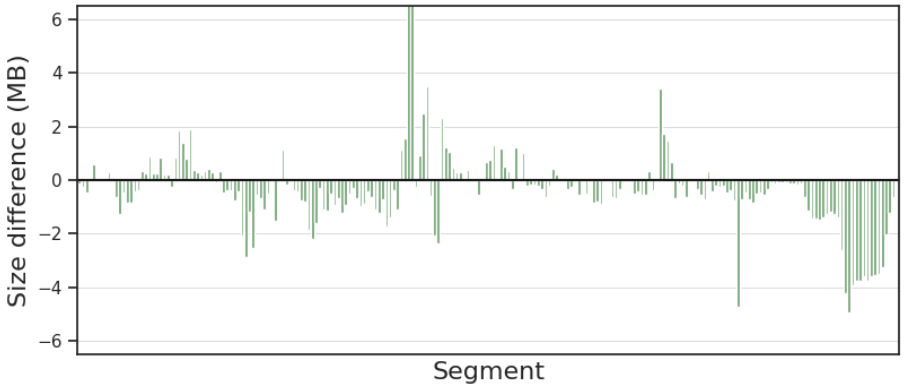
La Figura 3.11 compara el bitrate medio obtenido para dos esquemas de codificación (fijo y adaptativo) y tres vídeos (BBB, STL y TOS) cuando los segmentos se codifican a diferentes resoluciones. El esquema adaptativo propuesto presenta un bitrate inferior para todas las resoluciones y vídeos.

Por último, la Tabla 3.8 proporciona información sobre los resultados obtenidos para los dos esquemas comparados (adaptativo vs. fijo). Los datos obtenidos con nuestra propuesta se han comparado con una codificación con un valor de CRF fijo de 27, tal y como se indica en la subsección 3.3.2.2. Con este CRF obtenemos valores muy similares del valor objetivo de calidad elegido, en este caso VMAF. Para cada vídeo de entrada en bruto 4K se producen cinco vídeos codificados a diferentes resoluciones. Para cada resolución se proporcionan tres métricas (VMAF, SSIM y PSNR) junto con su desviación estándar para comprobar que nuestra propuesta funciona de forma correcta. También se muestra el tiempo de codificación del vídeo para cada resolución y su tamaño final.

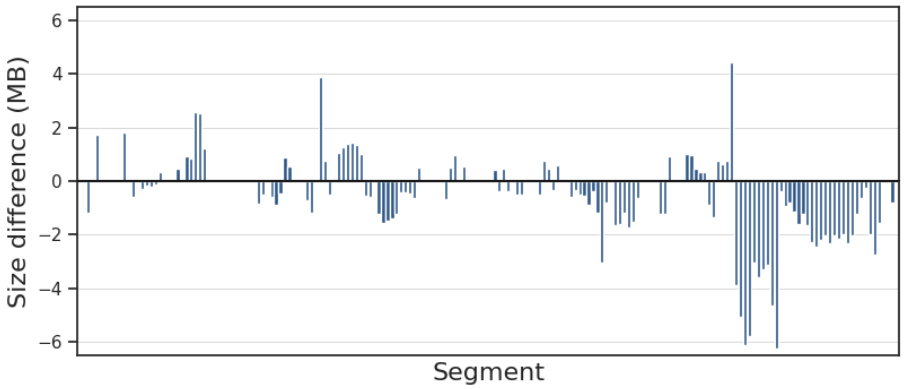
La Tabla 3.8-BBB muestra los datos obtenidos al aplicar nuestro sistema de codificación basado en la calidad sobre la película BBB. En este caso, podemos observar que nuestro sistema disminuye el tamaño de cada vídeo/-



(a) BBB vídeo



(b) STL vídeo



(c) TOS vídeo.

Figura 3.9: Diferencia de tamaño por segmento entre la codificación adaptativa y la fija para la resolución 4K.

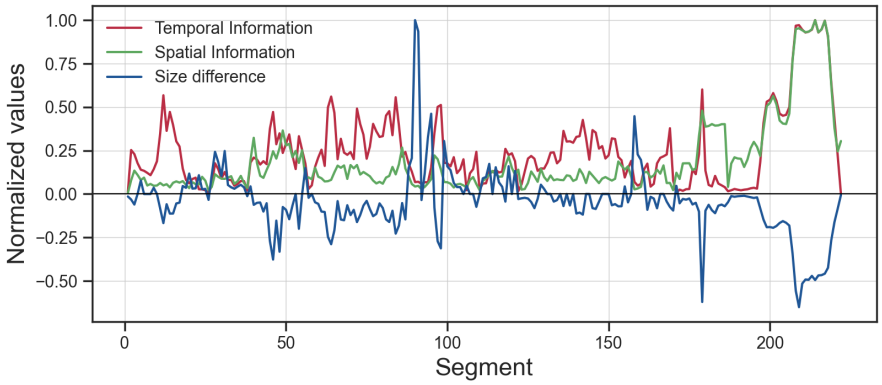


Figura 3.10: SI, TI y diferencia de tamaño normalizados para STL a una resolución de 4K.

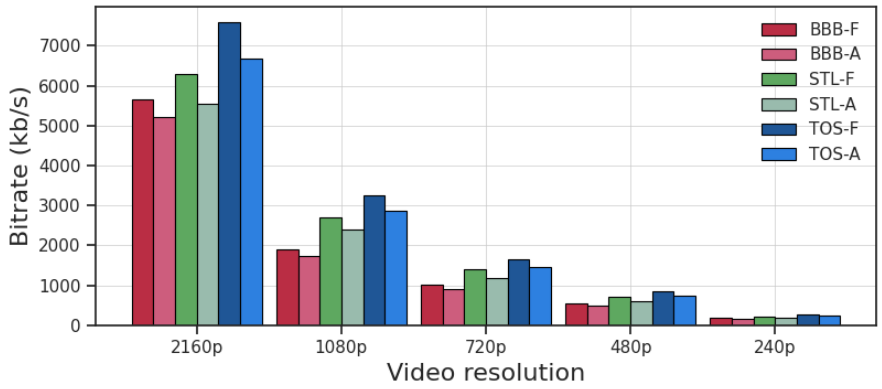


Figura 3.11: Tasa de bits para cinco resoluciones en los dos esquemas de codificación (fijo y adaptativo) y tres vídeos.

resolución respecto a la codificación con CRF fijo manteniendo un VMAF superior a 90 (nuestro valor objetivo). Además, podemos ver como la desviación estándar de nuestra propuesta es menor, lo que indica que en todos los segmentos estamos seleccionando el CRF más ajustado para obtener un vídeo con la métrica objetivo. Nuestra propuesta proporciona una mejora en tamaño que oscila entre el 7.8 % y el 9.9 %, con un valor medio de mejora del 8.4 %. En términos de tiempo, tenemos una mejora de 2.5 minutos sumando todos los tiempos requeridos para todas las resoluciones. Finalmente, todos los procesos de codificación han sido comprobados con VMAF, SSIM y PSNR y en todos los casos los datos obtenidos se consideran adecuados para proporcionar una buena QoE en términos de codificación de vídeo.

Cuando se analiza la película STL (ver Tabla 3.8-STL), los datos ob-

tenidos tienen una correlación similar con los resultados de BBB, pero con una ligera diferencia debido a que nuestra propuesta requiere 9 minutos más que utilizando un CRF fijo. En este caso, en todas las resoluciones tenemos valores de calidad por encima del valor objetivo de VMAF de 90. El porcentaje de mejora en tamaño oscila entre el 11.7 % y el 15.2 %, siendo el valor medio para todas las resoluciones del 12.3 %. Si analizamos los valores de VMAF, SSIM y PSNR, los vídeos codificados tienen valores altos y también las desviaciones estándar de estos datos son menores en nuestro sistema, lo que indica un mejor ajuste del nivel de calidad en todos los segmentos.

Tabla 3.8: Resultados de la codificación adaptativa vs. fija BBB y STL.

Vídeo	Núm. Segmentos	Resolución	Métrica	CRF Adaptativo				CRF Fijo				Adaptativo vs. Fijo
				Media	*Std dev*	** dimensionalidad, análisis, codificación (min)*	*Tamaño (MB)*	*Media*	*Std dev*	*Codificación (min)*	*Tamaño (MB)*	*Tamaño (%)*
BBB	159	240	VMAF	90.040	0.561	29.259	12.230	91.444	3.238	25.601	13.480	9.3
			SSIM	0.963	0.014			0.967	0.018			
			PSNR	37.408	2.963			38.239	3.492			
		480	VMAF	91.602	3.405	27.029	37.074	92.506	3.430	26.042	40.918	9.4
			SSIM	0.988	0.006			0.989	0.006			
			PSNR	38.973	3.944			39.733	3.742			
		720	VMAF	92.052	3.643	27.030	68.676	92.900	3.426	27.882	76.199	9.9
			SSIM	0.994	0.003			0.994	0.003			
			PSNR	40.130	3.957			40.867	3.759			
		1080	VMAF	92.439	3.790	27.954	131.328	93.254	3.432	27.668	144.102	8.9
			SSIM	0.996	0.002			0.996	0.002			
			PSNR	41.523	3.818			42.255	3.622			
		2160	VMAF	95.899	2.559	35.064	394.711	96.517	2.259	41.510	428.199	7.8
			SSIM	0.999	0.001			0.999	0.001			
			PSNR	44.355	3.951			45.130	3.778			
				TOTAL		146.336	644.020			148.703	702.898	8.4
STL	222	240	VMAF	90.076	0.597	23.435	19.950	91.863	4.787	17.340	22.941	13.0
			SSIM	0.959	0.026			0.968	0.018			
			PSNR	38.864	5.052			39.809	4.730			
		480	VMAF	91.384	3.749	19.027	63.090	92.614	4.783	18.429	74.355	15.2
			SSIM	0.984	0.010			0.988	0.007			
			PSNR	40.309	5.593			41.333	4.625			
		720	VMAF	91.829	4.118	20.015	126.648	92.786	4.814	19.778	149.184	15.1
			SSIM	0.991	0.006			0.993	0.004			
			PSNR	41.026	5.556			41.942	4.714			
		1080	VMAF	91.983	4.371	26.821	254.090	92.847	4.880	26.363	286.617	11.3
			SSIM	0.994	0.004			0.995	0.003			
			PSNR	41.677	5.262			42.536	4.428			
		2160	VMAF	94.722	3.371	49.791	588.039	95.202	3.553	48.104	666.043	11.7
			SSIM	0.997	0.002			0.997	0.001			
			PSNR	42.166	5.025			42.941	4.262			
				TOTAL		139.089	1051.820			130.014	1199.14	12.3

Tabla 3.9: Resultados de la codificación adaptativa vs. fija en TOS.

Vídeo	Núm. Segmentos	Resolución	Métrica	CRF Adaptativo				CRF Fijo				Adaptativo vs. Fijo
				Media	*Std dev*	** dimensionalidad. análisis. codificación (mín)*	*Tamaño (MB)*	*Media*	*Std dev*	*Codificación (mín)*	*Tamaño (MB)*	*Tamaño (%)*
TOS	184	240	VMAF	90.060	0.744	19.025	21.630	91.817	4.476	13.763	24.359	11.2
			SSIM	0.956	0.019			0.962	0.018			
			PSNR	36.398	3.653			37.269	4.149			
		480	VMAF	91.323	3.584	15.401	65.410	92.521	4.410	15.374	74.070	11.7
			SSIM	0.983	0.009			0.985	0.007			
			PSNR	38.071	4.528			38.961	4.068			
		720	VMAF	91.504	3.979	16.408	126.887	92.544	4.482	16.518	145.449	12.8
			SSIM	0.990	0.005			0.991	0.004			
			PSNR	39.058	4.155			39.940	3.708			
		1080	VMAF	91.115	4.366	22.366	250.582	92.127	4.719	29.054	284.902	12.0
			SSIM	0.992	0.004			0.993	0.003			
			PSNR	39.635	3.943			40.414	3.462			
		2160	VMAF	93.172	3.748	35.896	584.703	93.780	3.912	41.369	665.383	12.1
			SSIM	0.996	0.003			0.996	0.002			
			PSNR	39.382	4.272			40.109	3.857			
			TOTAL			109.096	1049.210			116.078	1194.163	12.14

En Table 3.9-TOS tenemos la misma información pero para el vídeo TOS. En este caso, nuestra propuesta nos da beneficios en términos de tamaño y tiempo de codificación. Todas las resoluciones en el proceso de codificación basado en la calidad cumplen la métrica objetivo (VMAF≥90). Los porcentajes de mejora en espacio en disco de los vídeos codificados con nuestro sistema oscilan entre el 11,2 % y el 12,8 %, con una mejora media del 12,1 %. Si nos fijamos en el tiempo, la codificación con CRF fijo para las 5 resoluciones tarda 116 minutos y con CRF adaptativo 109 minutos, obtenemos una mejora de 7 minutos. Lo mismo ocurre con todas las métricas de calidad (VMAF, SSIM y PSNR) que en los casos anteriores.

Para resumir esta sección, podemos decir que nuestro sistema de codificación multirresolución basado en la calidad es capaz de disminuir el tamaño de los vídeos en más de un 10 % utilizando el mismo tiempo de codificación. Además, con este sistema propuesto, los vídeos salientes siempre tienen un valor de calidad igual o superior al indicado como objetivo y este valor de calidad tiene una desviación estándar menor con respecto a una codificación CRF fija.

3.4. Conclusiones

En HAS, la preparación del contenido es muy importante, ya que debe maximizar la eficiencia en el proceso de codificación (calidad frente a tamaño y tiempo). Para mejorar este aspecto, en este capítulo se han presentado dos aproximaciones de codificación adaptativa basada en la calidad para preparar el contenido de vídeo para una transmisión VOD con DASH. En los dos casos, cada segmento de un vídeo sin procesar se codifica para proporcionar una calidad objetivo. Para cada segmento se resuelve un problema de optimización, en el primer esquema, se utiliza una interpolación para identificar el valor de CRF que se ajuste a la calidad deseada; en segundo esquema, se utiliza el descenso de gradiente con un paso adaptativo para obtener el valor de este parámetro. Este análisis se realiza a baja resolución para mejorar el rendimiento —240p en los dos casos—. En el primer esquema, cada segmento del vídeo reducido es codificado en un rango fijo predefinido de CRF (8 interacciones), por otra parte, el segundo esquema, en promedio, usa 3 interacciones antes de converger.

Los dos esquemas se han comparado con la utilización de un valor de CRF fijo por segmento. En la sección de análisis, el primer esquema muestra

los resultados utilizando diferentes tamaños de segmento a una resolución 1080p, partiendo de dos vídeos de alta resolución (4k) sin procesar. En el segundo caso, se utilizan tres vídeos y se producen 5 resoluciones por vídeo con un tamaño de segmento de 4 segundos.

Los experimentos con dos vídeos y diferentes tamaños de segmento muestran que el primer esquema puede reducir el tamaño del vídeo codificado en torno al 10 % en comparación con un CRF fijo por segmento. Por otra parte, los resultados del segundo esquema muestran que la codificación adaptativa propuesta alcanza la calidad objetivo con una reducción de entre el 8 % y el 12 % del tamaño de los vídeos codificados.

Capítulo 4

Mejora de la codificación DASH mediante análisis de escenas y técnicas de reducción de escala para la transmisión VOD

La simplicidad es la máxima sofisticación.

Leonardo da Vinci

Resumen:

En este capítulo se estudian los efectos a nivel de tiempo, calidad y tamaño de los vídeos codificados cuando se utiliza una reducción de la resolución del vídeo para obtener sus escenas. Los experimentos se realizan utilizando dos códecs: H.264 y VP9, y utilizando 10 vídeos con resolución 4K con una duración superior a 150 segundos. Los vídeos se codifican utilizando segmentos fijos y segmentos variables basados en escenas — obtenidas a partir del vídeo redimensionado —. Los resultados muestran que el uso de la reducción de la dimensionalidad (o *downscaling*) para obtener las escenas tiene un impacto mínimo en la calidad final, a la vez que reduce el tiempo total, el consumo de recursos computacionales y el tamaño del vídeo codificado. Este capítulo está respaldado por la publicación (Moina-Rivera et al., 2021c).

4.1. Introducción

Hay dos elementos en los sistemas DASH que son clave para ofrecer una experiencia de alta calidad a los usuarios finales. Por un lado, el contenido debe prepararse en el servidor, es decir, seleccionar los códecs y representaciones adecuados con los ajustes correctos para ofrecer una buena calidad y, al mismo tiempo, minimizar la tasa de bits para mejorar la transmisión del contenido multimedia y su almacenamiento.

Por otro lado, el algoritmo de selección de segmentos DASH implementado por el cliente. Este algoritmo se encarga de seleccionar el mejor segmento para evitar efectos negativos como eventos de bloqueo en el *streaming* de vídeo.

En el *streaming* DASH, el proceso de codificación puede ser tan importante como la selección del segmento adecuado en el lado del cliente. En este primer proceso, la resolución, la tasa de bits o el tamaño del segmento, entre otros parámetros, deben seleccionarse para conseguir la máxima calidad. La elección de estos parámetros también puede tener un impacto significativo en el rendimiento del codificador en términos de uso de recursos, calidad y tasa de bits.

Por estas razones, proponemos una solución de codificación basada en la segmentación de escenas de duración variable obtenidas a partir de la versión del vídeo a baja resolución. Medimos el impacto de este enfoque en términos de tiempo total (para obtener las escenas y codificarlas a diferentes resoluciones), calidad y tamaño de los vídeos codificados.

El esquema propuesto ha sido probado con dos codificadores, H.264 y VP9, sobre 10 vídeos[1]. Los resultados muestran que se ahorran recursos computacionales y que la calidad del vídeo codificado apenas se ve afectada a nivel de métrica, siendo esta diferencia imperceptible por el ojo humano.

Este capítulo está organizado como sigue. En la Sección 2 se presenta nuestro sistema de codificación de vídeo basado en escenas y técnicas de *downscaling*. La Sección 3 muestra varias pruebas de nuestra propuesta con el fin de validarla. Finalmente, en la Sección 4 se muestran las conclusiones.

[1]Una demo está disponible en: https://links.uv.es/jgutierr/scncoding

4.2. Codificación basada en escenas y técnicas de reducción de escala para DASH

Como es sabido, en HAS el vídeo se divide en segmentos no solapados que pueden ser de duración fija o variable. Si prestamos atención a la segmentación variable, necesitamos un primer paso para determinar dónde hay cambios de escena. Esto se puede realizar codificando el vídeo con FFmpeg usando x264 con detección de escena (esta es la configuración por defecto) y detectando las posiciones de los fotogramas clave ,*keyframes*.

A cada secuencia de entrada `video_in.mov` se le aplica una transcodificación para cambiar el formato de 10 bits 4:2:2 a 8 bits 4:2:0 sobre un contenedor MP4 (`video.mp4`). Este cambio se ha realizado para hacer el streaming más compatible con la mayoría de las aplicaciones que implementan HAS y con los navegadores web disponibles en el mercado. Para este proceso se ha utilizado la herramienta FFmpeg con la instrucción número 1 que aparece en la Tabla 4.1. Para determinar el impacto de la resolución en la detección de escenas en la codificación DASH, hemos planificado una solución basada en tres pasos (ver Figura 4.1), que se describen a continuación.

Tabla 4.1: Ejemplos de instrucciones para cada etapa

1	`ffmpeg -i  video_in.mov -pix_fmt yuv420p -c:v libx264 -crf 0 -g 1 -an video.mp4`
2	`ffmpeg -i video.mp4 -c:v libx264 -vf scale="-2:resolution" \` `-threads 16 -g ((fps*10)) video_scenes.mp4`
3	`ffprobe -select_streams v:0 -show_entries packet=pts_time,flags \` `-of csv=print_section=0 video_scenes.mp4 | awk -F',' '/K/ {print $1}'`
4	`ffmpeg -ss time_i video.mp4 -time_f -c:v libx264 -crf crf -threads 16 scene_j.mp4`
5	`ffmpeg -ss time_i -i video.mp4 -t time_f -c:v codec -crf crf -b:v 0 -threads 16 \` `-speed 2 -quality good -tile-columns 6 -frame-parallel 1 scene_j.mp4`
6	`MP4Box -dash 2000 -frag 2000 -rap video.mp4`

4.2.1. Reducción de escala y detección de escenas

Dado que esta propuesta utiliza FFmpeg como herramienta para la detección de escenas, se toman varias consideraciones y restricciones sobre la

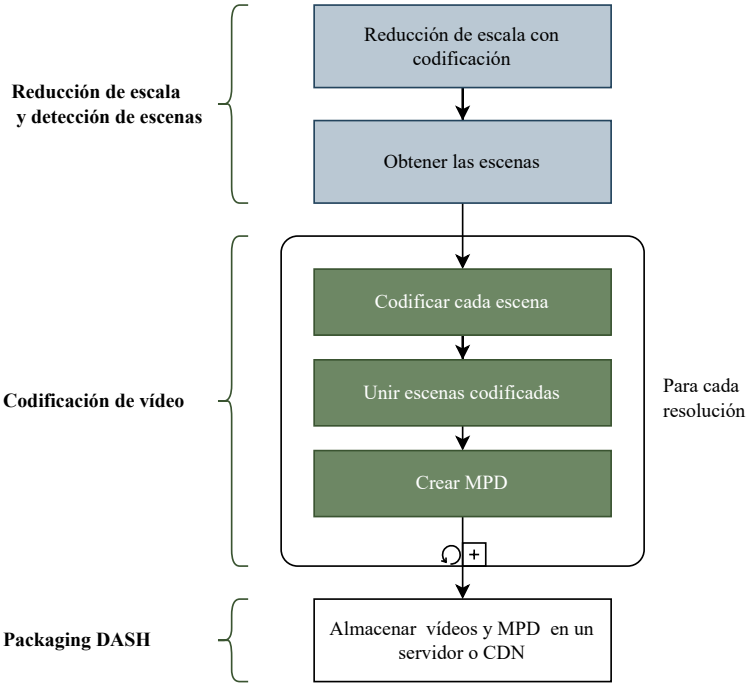


Figura 4.1: Propuesta de *pipeline* de codificación basada en el *downscaling* para obtener las escenas.

codificación de vídeo que se utilizará para extraer la información sobre los tiempos de inicio y fin de cada segmento. El códec utilizado para la detección de escenas es x264, porque implementa por defecto la detección de cambios de escena y por su compromiso entre codificación y calidad temporal. En las pruebas realizadas se ha utilizado el valor por defecto (40) para el parámetro umbral de detección de escenas, `sc_threshold`. Es por ello, que dicho parámetro no sé indica en la instrucción número 1 de la Tabla 4.1.

Debido a las características y limitaciones de los algoritmos adaptativos implementados en el estándar DASH, se especifica un tamaño de segmento con un mínimo de 2s y un máximo de 10s. El valor máximo se establece para evitar efectos de *buffering* en la reproducción y el mínimo para evitar cuellos de botella en las peticiones de los servidores implicados. Primero se reduce la escala del vídeo, mediante un algoritmo bicúbico, manteniendo la relación de aspecto del vídeo original y luego se codifica con las consideraciones anteriores. La instrucción utilizada en FFmpeg se muestra en la fila número 2 de la Tabla 4.1. El parámetro de tamaño máximo del GOP (`-g`) se calcula en función de los fps del vídeo de origen. Utilizando como entrada `video_scenes.mp4` en la herramienta `ffprobe`, se extrae la posición de cada

keyframe, que delimita el principio y el final de cada escena. Por tanto, las escenas se obtienen a partir del vídeo reescalado y codificado. La instrucción utilizada se muestra en la fila número 3 de la Tabla 4.1.

Para ajustar el tamaño mínimo del segmento, las escenas con un intervalo de tiempo inferior a 2s se añaden a la escena siguiente sólo si la suma es menor o igual a 10s; se añadirán tantas escenas como sea necesario para ajustarse al mínimo especificado.

4.2.2. Codificación de vídeo

La entrada (`video.mp4`) se divide utilizando las marcas de tiempo de las escenas obtenidas en el paso anterior y se codifica cada escena. En este proceso, los segmentos se codifican con un valor fijo del parámetro CRF, ya que éste es el ajuste de calidad por defecto para codificadores como x264 y VP9. El rango de CRF para x264 está entre 0 y 51 y para VP9 entre 0 y 63, donde los valores más bajos proporcionan mejor calidad, pero mayores tasas de bits. La instrucción utilizada para codificar un segmento de vídeo con x264 se muestra en la fila número 4 de la Tabla 4.1, donde `time_i` es el tiempo inicial de una escena y `time_f` es el tiempo final de la misma escena. La instrucción utilizada para codificar un segmento de vídeo con VP9 se muestra en la fila número 5 de la Tabla 4.1.

Como puede verse en estos ejemplos, el vídeo se codifica a la resolución original pero utilizando las escenas obtenidas del vídeo reducido.

4.2.3. Empaquetado DASH

Por último, una vez que tenemos los vídeos codificados por escenas, necesitamos empaquetarlos correctamente para crear el archivo MPD. Este archivo, junto con los vídeos, se almacenará en el servidor HTTP para permitir el streaming DASH desde un reproductor web. Este último paso se ha realizado con las herramientas GPAC[2] tal y como se muestra en la fila número 6 de la Tabla 4.1, donde `-dash` habilita la segmentación DASH de los archivos de entrada con la duración de segmento dada, `-frag` especifica la duración del segmento en ms y `-rap` fuerza a que los segmentos comiencen con puntos de acceso aleatorios.

[2]https://github.com/gpac/gpac

Tabla 4.2: Vídeos con una resolución de 3840x2160 utilizados en los experimentos.

Video	FPS	Frames	Size (GB)
Ancient thought (ANC)	23.98	4486	8.3
El dorado (ELD)	23.98	4384	17
Indoor soccer (IND)	23.98	6035	23
Liftingoff (LIF)	23.98	4825	15
Seconds that count (SEC)	23.98	7722	17
Skateboarding (SKA)	23.98	6607	23
Unspoken friend (UNS)	23.98	6166	18
Lifeuntouched (LIE)	59.94	11842	29
Moment of intensity (MOM)	59.94	11394	38
Wipe HDR (WIP)	59.94	9353	24

4.3. Experimentos y resultados

En esta sección presentamos nuestro entorno de pruebas (o *test bed*), los experimentos y los resultados que permiten validar nuestra propuesta.

4.3.1. Entorno de pruebas

En este trabajo, los vídeos se han codificado utilizando x264 y VP9 por ser los códecs que ofrecen mejor compatibilidad en entornos DASH. Para evaluar y validar el método propuesto, se han realizado varias pruebas sobre 10 vídeos[3], cuyas características se muestran en la Tabla 4.2. Para comprobar la validez de la propuesta independientemente de las características del vídeo, los vídeos seleccionados tienen una amplia variedad de características de Información Espacial vs. Información Temporal (SI-TI), como se muestra en la Figura 4.2.

El hardware/software utilizado en todos los experimentos se especifica en la Tabla 4.3. La máquina física, ver Figura 4.3, tiene un hipervisor KVM que aloja una máquina virtual (VM). Los vídeos se almacenan en dos discos que

[3]https://www.cablelabs.com/4k

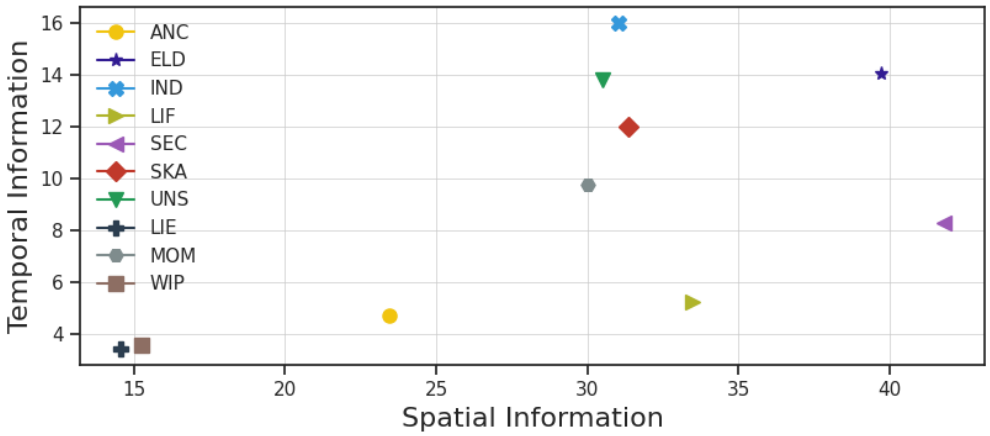


Figura 4.2: Información espacial promedio (SI) e información temporal (TI) de los vídeos utilizados.

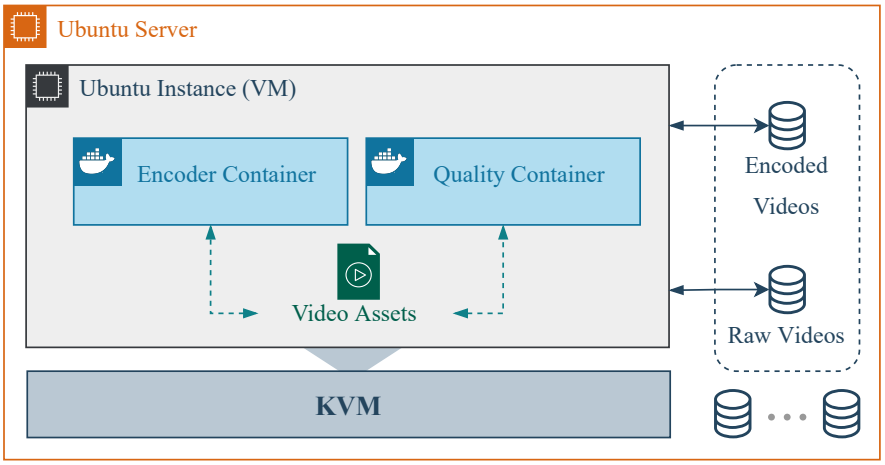


Figura 4.3: Arquitectura del *test-bed* con máquinas virtuales y contenedores.

están conectados a la VM mediante PCI-passthrough. La VM cuenta con un Docker engine que ejecuta dos contenedores: el ***Encoder Container*** implementa el sistema de codificación con sus diversos componentes para la preparación del contenido en múltiples representaciones y configuraciones; el ***Quality Container*** se utiliza para el análisis de calidad con soporte para diferentes métricas.

Tabla 4.3: Características hardware del test-bed

Atributo	Detalles
	Máquina física
CPU	2 Intel(R) Xeon(R) Silver 4216. Cores per CPU: 16. Threads per core: 2, @ 2.10GHz
Cache size	L1: 1MiB, L2: 16 MiB, L3: 22 MiB
Motherboard	Intel S2600STB
RAM	128 GiB
OS	Ubuntu 20.04.2 LTS x64 bits
Hypervisor	Kernel-based Virtual Machine (KVM)
	Máquina virtual
CPU	24 (1 thread)
RAM	64 GiB
Storage	HGST HUS722T2TAL 2 TB 7.200 RPM ST2000DM008-2FR1 2TB 7.200 RPM
OS	Ubuntu 20.04.2 LTS x64 bits
Docker	Version 20.10.5

4.3.2. Caso base

En este experimento se han codificado todos los vídeos 4K con formato 8-bit 4:2:0 con VP9 y x264 utilizando valores CRF que producen una calidad similar a los vídeos codificados.

Todos los vídeos han sido codificados forzando un *keyframe* cada 4 segundos con el fin de tener los vídeos preparados para ser transmitidos en DASH con un tamaño de segmento fijo de 4 segundos. Esta selección se ha realizado siguiendo las recomendaciones de (Lederer, 2015).

En la Tabla 4.4 se muestra el resumen de estas pruebas, donde aparecen varias métricas de calidad, el tamaño en MB y el tiempo de codificación del vídeo en minutos. Estos valores se obtienen para un CRF=31 para x264 y un CRF=47 para VP9 (nótese que el rango de la escala CRF es diferente para cada codificador: $\{0-51\}$ para x264 y $\{0-63\}$ para VP9).

Si nos fijamos en las últimas filas, donde se muestran los valores medios,

Tabla 4.4: Tiempo, calidad y tamaño de los vídeos de prueba codificados con x264 (CRF=31) y VP9 (CRF=47).

Vídeo	**CRF - Códec**	**VMAF**	**SSIM**	**PSNR**	**Tamaño (MB)**	**Tiempo (min)**
ANC	31 - x264	92.637	0.997	47.644	35.216	5.373
	47 - vp9	91.610	0.997	47.654	10.840	16.604
ELD	31 - x264	88.708	0.994	39.294	158.452	4.618
	47 - vp9	87.564	0.992	39.170	98.496	31.343
IND	31 - x264	89.622	0.992	40.170	203.480	7.482
	47 - vp9	89.486	0.991	40.173	99.144	38.907
LIF	31 - x264	90.189	0.997	42.659	74.444	4.013
	47 - vp9	89.413	0.996	42.690	44.188	27.159
SEC	31 - x264	93.484	0.997	45.008	97.304	6.142
	47 - vp9	92.593	0.996	44.994	45.984	35.741
SKA	31 - x264	91.063	0.995	41.321	128.280	6.940
	47 - vp9	89.994	0.994	41.210	65.524	36.197
UNS	31 - x264	89.822	0.995	42.156	145.712	5.739
	47 - vp9	89.313	0.994	42.242	72.580	35.214
LIE	31 - x264	86.164	0.995	44.951	81.696	10.711
	47 - vp9	87.490	0.995	45.637	36.352	51.642
MOM	31 - x264	84.610	0.985	39.568	118.460	13.361
	47 - vp9	88.619	0.990	41.113	89.620	61.855
WIP	31 - x264	85.041	0.995	44.553	65.984	7.355
	47 - vp9	86.758	0.996	45.526	37.640	41.877
Mean Values	31 - x264	89.134	0.994	42.732	110.903	7.173
	47 - vp9	89.284	0.994	43.041	60.037	37.654

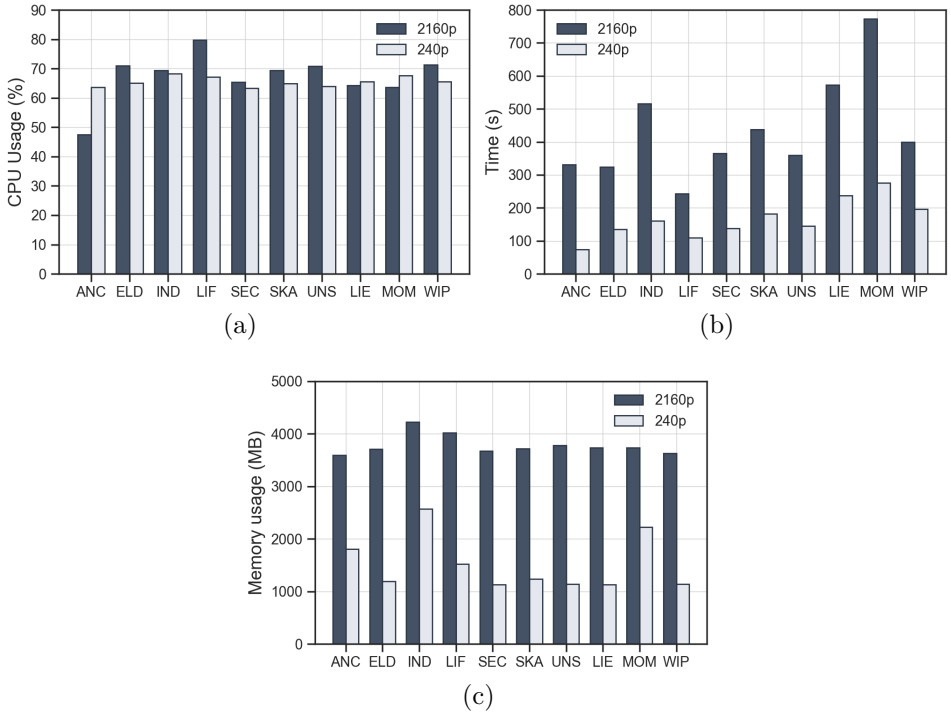


Figura 4.4: Recursos de hardware utilizados en la detección de escenas con/sin reducción de escala previa: a) CPU, b) tiempo de cómputo y c) memoria.

podemos ver que el valor medio de VMAF es superior a 89, el valor medio de SSIM es 0.994 y el valor de PSNR es superior a 42.5 dB. Todos estos valores indican que los vídeos codificados tienen una calidad muy buena.

En el caso de los resultados de tamaño y tiempo observamos que el códec VP9 ofrece una reducción del tamaño del 45 %, pero el tiempo de codificación se incrementa en más de un 500 %.

En el resto de los experimentos, en los que trabajamos con vídeos 4K, utilizamos estos valores CRF para codificar nuestros vídeos de prueba con un valor VMAF superior a 89, lo que implica tener un MOS superior a 4 (Li et al., 2018b).

4.3.3. Efecto de la resolución en la detección y codificación de escenas

En esta subsección, analizamos cómo afectan las técnicas de *downscaling* a la detección de escenas en dos aspectos: a) nivel de recursos hardware utilizados y b) variables en la codificación de vídeo 4K por escena (tiempos, tamaños, calidades). En primer lugar, la Figura 4.4 muestra los requisitos computacionales necesarios para realizar la detección de escenas a una resolución de 240p y en la resolución original de los vídeos de entrada (2160p).

La memoria RAM utilizada (ver Figura 4.4c) cuando se realiza la detección de escenas en el vídeo reescalado se reduce en un factor 2.51 respecto a realizar la misma tarea en resolución 4K. Un comportamiento similar se observa si analizamos el tiempo necesario para detectar las escenas (ver Figura 4.4b). Podemos ver que si este proceso se realiza a baja resolución el tiempo disminuye en un factor 2.61.

Respecto a los requerimientos de CPU, observamos (ver Figura 4.4a) que la detección de escenas con *downscaling* requiere menos CPU en la mayoría de las secuencias de vídeo. El codificador tiende a utilizar todos los núcleos disponibles en ambos casos, pero mucho menos tiempo con la resolución de 240p. Esto es importante si el sistema se despliega en una Nube pública con un modelo de pago por uso. Con el *downscaling* se necesita menos memoria y se reduce el tiempo de uso de los recursos.

A continuación, analizamos el impacto de la detección de escenas a baja resolución en las métricas de calidad, tiempos de codificación y tamaños de vídeo (ver Tabla 4.6). Esta tabla muestra la información anterior para dos codecs (x264 y VP9) cuando la detección de escenas se realiza con una resolución baja (240p) o con la resolución original (2160p).

El primer aspecto a analizar es el número de escenas detectadas con ambas resoluciones. En términos generales, podemos decir que la diferencia es muy baja, excepto en aquellos vídeos con valores SI-TI bajos, donde la resolución 2160p detecta algunas escenas más. De media, se detectan 2.4 escenas menos cuando se aplican técnicas de *downscaling* . Pero esta diferencia de escenas no afecta en términos de calidad, ya que todos los vídeos tienen valores VMAF, SSIM y PSNR similares en ambos casos.

Observando los tiempos de codificación para el códec x264, la mejora obtenida en una codificación basada en escenas aplicando *downscaling* en la

fase de detección es superior al 28 %. Teniendo valores superiores al 38 % en algunos vídeos y nunca inferiores al 21 %, lo que puede reducir los costes por uso en infraestructuras en la Nube.

El tamaño del archivo, el número de escenas y la calidad son similares ya que los vídeos se codifican utilizando el mismo valor de CRF (31). Tenemos un comportamiento similar en términos de calidad y tamaño con el códec VP9. Cuando se aplica una detección de escenas de baja resolución, el tiempo total de codificación también disminuye, pero en este caso el porcentaje es menor. Esto se debe a que el códec VP9 requiere un mayor tiempo de codificación. Con VP9 hay una mejora media de más del 9 %, llegando en algunos vídeos a más del 18 % y en el peor de los casos a más del 5 %.

4.3.4. Codificación por escenas frente a codificación fija

En esta subsección, comparamos la codificación basada en escenas utilizando la versión reducida del vídeo a 240p de resolución para obtener las escenas, frente a la codificación fija por segmentos en escenarios DASH. La Figura 4.5 muestra los resultados para ambos códecs — arriba los resultados para x264 y abajo para VP9 —. Las figuras 4.5a y 4.5b muestran que utilizando el códec x264 el VMAF obtenido con la codificación por escenas es ligeramente inferior al de la codificación por segmentos fijos de 4 segundos, mientras que, en términos de tamaño, el vídeo codificado por escenas ocupa menos. En términos medios, estaríamos hablando de una diferencia de calidad del 0.6 % pero tenemos un ahorro de tamaño del 7 %. Estos resultados mejoran con el códec VP9 (ver figuras 4.5c y 4.5d), donde la calidad es igual o ligeramente superior usando codificación por escenas respecto a la codificación por segmentos fijos. Además, en todos los vídeos hemos obtenido un valor medio de ahorro de tamaño del 5.8 %.

4.3.5. Codificación multirresolución

En esta subsección se han codificado cuatro vídeos (ELD, IND, SEC, WIP) en cinco resoluciones utilizando cinco CRFs con x264 (ver Tabla 4.7). Estas codificaciones se han realizado utilizando segmentos fijos de tamaño 4 segundos y tamaños de segmento basados en escenas (obtenidas del vídeo de baja resolución). En la Tabla 4.7, podemos observar los porcentajes medios

Tabla 4.5: Efecto de la reducción de escala para la detección de escenas en el tiempo, la calidad y el tamaño de los videos codificados para x264.

Vídeo	Resoluciones		Métrica Calidad	x264					
	240p	*2160p*		*240p*	*2160p*	*240p*	*2160p*	*240p*	*2160p*
	Número de escenas			*Calidad media*		*Tiempo total (min)*		*Tamaño (MB)*	
ANC	30	27	VMAF	92.239	92.278	7.180	11.620	33.371	34.312
			SSIM	0.997	0.997				
			PSNR	47.040	47.049				
ELD	27	27	VMAF	88.271	88.347	8.070	11.010	144.254	144.301
			SSIM	0.994	0.994				
			PSNR	38.970	39.019				
IND	52	52	VMAF	89.158	89.179	12.000	17.710	191.676	191.996
			SSIM	0.992	0.992				
			PSNR	39.733	39.724				
LIF	27	27	VMAF	89.667	89.667	6.990	8.956	68.906	68.832
			SSIM	0.996	0.996				
			PSNR	42.299	42.310				
SEC	59	60	VMAF	92.710	92.742	10.190	14.120	89.410	89.617
			SSIM	0.997	0.997				
			PSNR	44.133	44.170				
SKA	44	50	VMAF	90.437	90.382	11.060	15.550	117.848	117.371
			SSIM	0.995	0.995				
			PSNR	40.834	40.784				
UNS	39	39	VMAF	89.323	89.312	9.280	12.770	136.625	136.605
			SSIM	0.995	0.995				
			PSNR	41.725	41.724				
LIE	36	45	VMAF	85.705	85.659	15.410	20.890	77.730	77.996
			SSIM	0.995	0.995				
			PSNR	44.693	44.688				
MOM	38	40	VMAF	84.109	84.094	19.600	27.350	108.434	108.402
			SSIM	0.984	0.984				
			PSNR	39.253	39.277				
WIP	22	31	VMAF	84.541	84.501	10.950	14.630	62.137	62.188
			SSIM	0.995	0.995				
			PSNR	44.266	44.260				

Tabla 4.6: Efecto de la reducción de escala para la detección de escenas en el tiempo, la calidad y el tamaño de los vídeos codificados para VP9.

Vídeo	**Resoluciones**		**Métrica Calidad**	**VP9**					
	240p	***2160p***		***240p***	***2160p***	***240p***	***2160p***	***240p***	***2160p***
	Número de escenas			***Calidad media***		***Tiempo total (min)***		***Tamaño (MB)***	
ANC	30	27	VMAF	91.557	91.495	17.910	21.920	10.406	10.375
			SSIM	0.997	0.997				
			PSNR	47.440	47.424				
ELD	27	27	VMAF	87.792	87.761	34.140	37.350	92.191	92.254
			SSIM	0.993	0.993				
			PSNR	39.096	39.109				
IND	52	52	VMAF	89.520	89.451	41.760	47.180	94.047	94.508
			SSIM	0.991	0.991				
			PSNR	40.171	40.122				
LIF	27	27	VMAF	89.461	89.469	29.500	30.280	41.930	41.992
			SSIM	0.996	0.996				
			PSNR	42.577	42.584				
SEC	59	60	VMAF	92.676	92.741	37.400	41.170	42.488	42.648
			SSIM	0.996	0.996				
			PSNR	44.711	44.866				
SKA	44	50	VMAF	90.385	90.451	38.920	42.850	61.422	62.129
			SSIM	0.995	0.995				
			PSNR	41.148	41.196				
UNS	39	39	VMAF	89.369	89.405	36.850	40.120	68.543	68.891
			SSIM	0.994	0.994				
			PSNR	42.190	42.217				
LIE	36	45	VMAF	87.516	87.518	56.080	61.780	34.480	34.770
			SSIM	0.995	0.995				
			PSNR	45.643	45.661				
MOM	38	40	VMAF	88.694	88.618	67.330	75.250	84.574	84.770
			SSIM	0.990	0.990				
			PSNR	41.064	41.032				
WIP	22	31	VMAF	86.700	86.773	44.050	46.550	35.676	35.973
			SSIM	0.996	0.996				
			PSNR	45.511	45.538				

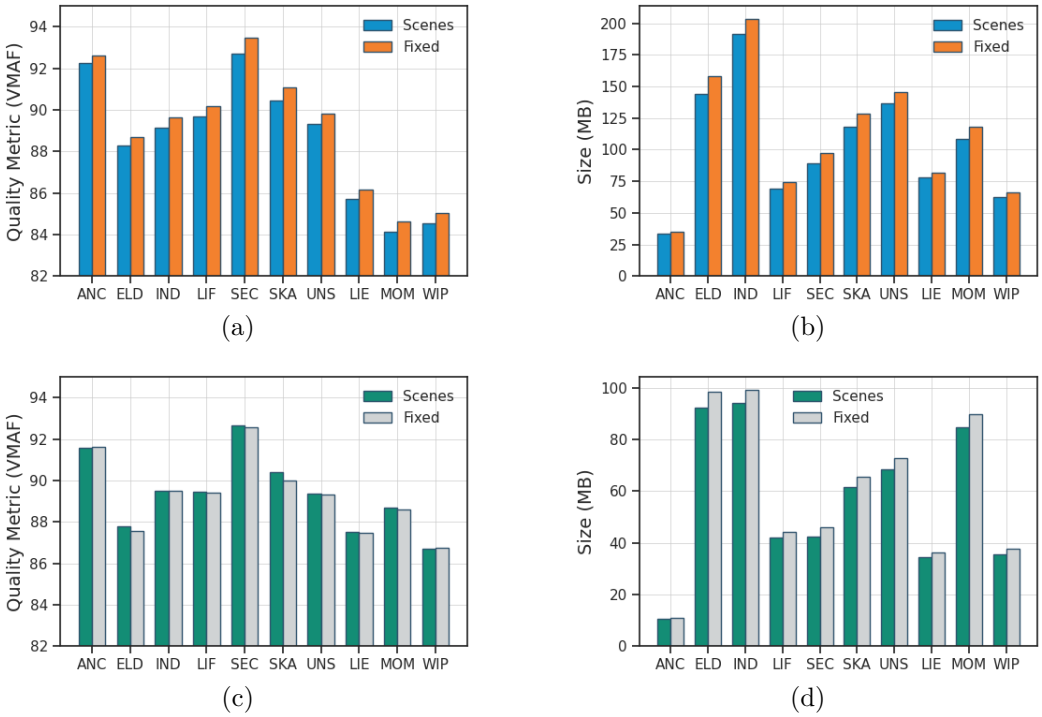


Figura 4.5: Comparación de calidad y tamaño de la codificación fija frente a la basada en escenas utilizando x264 (a y b) y VP9 (c y d).

sobre todos los vídeos ($\pm\Delta\,\%$) de SSIM, PSNR y tamaño para cada resolución y cada CRF. En este caso, los valores positivos significan una mejora para la codificación basada en escenas y los valores negativos indican una mejora para la codificación con segmentos fijos. En términos generales, podemos ver que tenemos una mejora media del 7.8 % en el tamaño del archivo y esta mejora sólo se traduce en una disminución media de la calidad del 0.28 % para el SSIM y del 1.61 % para el PSNR. Estos pequeños valores porcentuales no son percibidos por un usuario final, ya que estos porcentajes corresponden a una variación media de 0.003 en SSIM y 0.6dB en PSNR sobre los valores absolutos.

Tabla 4.7: Porcentaje promedio de ganancia/pérdida ($\pm\Delta$ %) para cuatro vídeos usando cinco CRF en cinco resoluciones.

Resoluciones	CRF=19			CRF=23			CRF=27			CRF=31			CRF=35		
	SSIM	***PSNR***	***Tamaño***	***SSIM***	***PSNR***	***Tamaño***	***SSIM***	***PSNR***	***Tamaño***	***SSIM***	***PSNR***	***Tamaño***	***SSIM***	***PSNR***	***Tamaño***
240p	-0.126	-1.862	7.047	-0.241	-2.015	8.477	-0.458	-2.284	9.690	-0.790	-2.480	10.437	-1.213	-2.590	10.476
480p	-0.073	-1.574	6.915	-0.159	-1.984	8.101	-0.297	-1.705	9.546	-0.544	-1.960	10.267	-0.869	-2.145	10.463
720p	-0.043	-1.349	6.506	-0.083	-1.400	8.063	-0.196	-1.585	9.057	-0.387	-1.762	9.547	-0.632	-1.849	9.728
1080p	-0.031	-1.208	5.522	-0.029	-1.201	6.943	-0.089	-1.314	7.903	-0.223	-1.493	8.052	-0.412	-1.678	8.009
2160p	-0.034	-0.783	3.765	-0.020	-0.815	4.952	0.001	-0.914	6.001	-0.020	-1.100	5.978	-0.101	-1.304	5.655

4.4. Conclusiones

Este capítulo propone una solución de codificación basada en escenas para mejorar la eficiencia del proceso de codificación DASH, que proporciona un mejor rendimiento en términos de uso de recursos, calidad y tasa de bits. Esta solución se basa en aplicar un agresivo *downscaling* al vídeo de entrada (pasando de 2160p a 240p) para detectar las escenas. Utilizando esta información, realiza una codificación por segmentos basada en las escenas obtenidas y prepara el vídeo final para ser enviado a través de DASH. Los resultados muestran que este *downscaling* no afecta a la calidad final de los vídeos codificados y proporciona los siguientes beneficios: a) reducción en el uso de recursos y tiempos medios totales de codificación, una mejora media del 28 % en x264 y del 9 % usando VP9 respecto a la detección de escenas a resolución completa; y b) reducción en el tamaño de los vídeos finales, en términos medios del 7 % con x264 y del 5.8 % con VP9.

Capítulo 5

Codificación de vídeo y evaluación de la calidad sobre entornos serverless

No he fracasado. Simplemente he encontrado 10,000 formas que no funcionan

Thomas Edison

Resumen:

Tal y como se ha indicado en capítulos anteriores, hoy en día, la mayor parte del tráfico de Internet son contenidos multimedia. Los servicios de *streaming* de vídeo son muy demandados por los usuarios finales y utilizan *HTTP Adaptive Streaming* como protocolo de transmisión. HAS divide el vídeo en trozos no solapados y cada trozo de vídeo puede codificarse de forma independiente utilizando diferentes representaciones. Por lo tanto, estas tareas de codificación pueden paralelizarse y utilizar todos los beneficios que puede aportar la computación en Nube. Sin embargo, en las soluciones más extendidas, la infraestructura debe configurarse y aprovisionarse de antemano. Recientemente, las plataformas sin servidor (o *serveless*) han hecho posible desplegar funciones que pueden escalar desde cero hasta un máximo configurable. Este capítulo presenta y analiza el comportamiento de funciones *serveless* dirigidas por eventos para codificar trozos de vídeo y calcular, opcionalmente, la calidad de los vídeos codificados. Estas funciones se han implementado utilizando una versión adaptada

de Tomcat embebido para tratar con CloudEvents. Hemos desplegado estas canalizaciones (o *pipelines) serverless* basados en eventos para la codificación y la evaluación de la calidad del vídeo en una plataforma *serveless* en local y basada en Knative con un nodo maestro (o *master*) y ocho nodos esclavos (o *workers*). Hemos probado la escalabilidad y el consumo de recursos de la solución propuesta utilizando dos códecs de vídeo: x264 y AV1, variando el número máximo de réplicas y los recursos asignados a las mismas (réplicas de función *fat* y *slim*). Además, se han codificado diferentes vídeos 4K para generar múltiples representaciones por cada llamada a la función y mostramos cómo es posible crear *pipelines* de funciones multimedia serverless. Los resultados de las diferentes pruebas realizadas muestran el buen rendimiento de las funciones serverless propuestas. El sistema escala las réplicas y distribuye los trabajos uniformemente entre todas ellas. El tiempo total de codificación se reduce en un 18 % utilizando réplicas *slim*, pero las réplicas *fat* son más adecuadas en el *streaming* de vídeo en directo, ya que se reduce el tiempo de codificación por segmento. Por último, los resultados de la prueba con *pipelines* muestran una distribución y una secuenciación adecuada de los trabajos entre las réplicas disponibles de cada tipo de función.

Este capítulo esta sustentado por el trabajo (Moina-Rivera et al., 2023a).

5.1. Introducción

Como hemos visto en los capítulos anteriores, en HAS, un vídeo se divide en segmentos temporales más pequeños y cada segmento se codifica con una resolución y tasa de bits diferente, creando varias representaciones. Por otra parte, los sistemas de *streaming* que los implementan requieren sistemas eficientes de codificación de vídeo para entregar contenidos en tantas resoluciones, tasas de bits y códecs como sea posible para proporcionar la mejor calidad de experiencia (QoE) a los usuarios finales Seufert et al. (2014).

Todos estos procesos de codificación y transcodificación requieren de sistemas distribuidos para mejorar los tiempos de codificación. Hoy en día, estas tareas se realizan en entornos de la Nube, ya que ofrecen el dinamismo que demandan las plataformas de *streaming* Gutiérrez-Aguado et al. (2020c). Las soluciones de codificación de vídeo en la Nube han evolucionado a medida que han surgido nuevos paradigmas en la Nube. Existen muchos servicios

en la Nube, todos ellos con el objetivo de ofrecer funcionalidades específicas adecuadas a un propósito. En concreto, son modelos de implementación de servicios que permiten al usuario elegir el nivel de control sobre la información y los servicios que van a proporcionar (véase la Figura 5.1). Los servicios más importantes que ofrecen los proveedores de la Nube son:

- IaaS (Infraestructura como servicio): En este modelo, el proveedor arrienda la infraestructura y el cliente tiene un control casi total sobre ella. El proveedor es responsable de garantizar la fiabilidad y seguridad a largo plazo mediante el mantenimiento de la infraestructura.
- PaaS (Plataforma como servicio): El proveedor proporciona el sistema de hardware y una plataforma de software de aplicación. PaaS permite al desarrollador desarrollar, ejecutar y gestionar sus aplicaciones sin tener que diseñar y mantener la infraestructura y la plataforma.
- SaaS (Software como servicio): Este modelo ofrece a los usuarios una aplicación específica que puede utilizarse inmediatamente, sin necesidad de instalar nada, desplegar nada ni mantenerla. Todo lo gestiona el proveedor del servicio.
- Serverless: Este tipo de arquitectura permite la ejecución de aplicaciones a través de contenedores efímeros. Estos contenedores se ejecutan sobre la marcha, por lo que el desarrollador no tiene que preocuparse de la gestión de la infraestructura sobre la que se ejecuta su función, centrándose exclusivamente en la funcionalidad. Suele englobar dos modelos de servicio complementarios: FaaS (*Function as a Service*) y BaaS (*Backend as a Service*).

En concreto, las arquitecturas *serverless* frente al resto de paradigmas de la Nube presentan las siguientes ventajas (+) y desventajas (−):

+ Delegar la administración del servidor: los tiempos de ejecución del programa o servicio se definen automáticamente, no requieren verificación constante ni instalación de *plugins*; simplemente el software se ejecuta automáticamente.
+ Proceso de escalado: Generalmente, el rendimiento y la capacidad de memoria requieren un proceso de escalado cuando se necesitan más recursos. Con la tecnología de computación sin servidor, este proceso se puede proporcionar automáticamente.

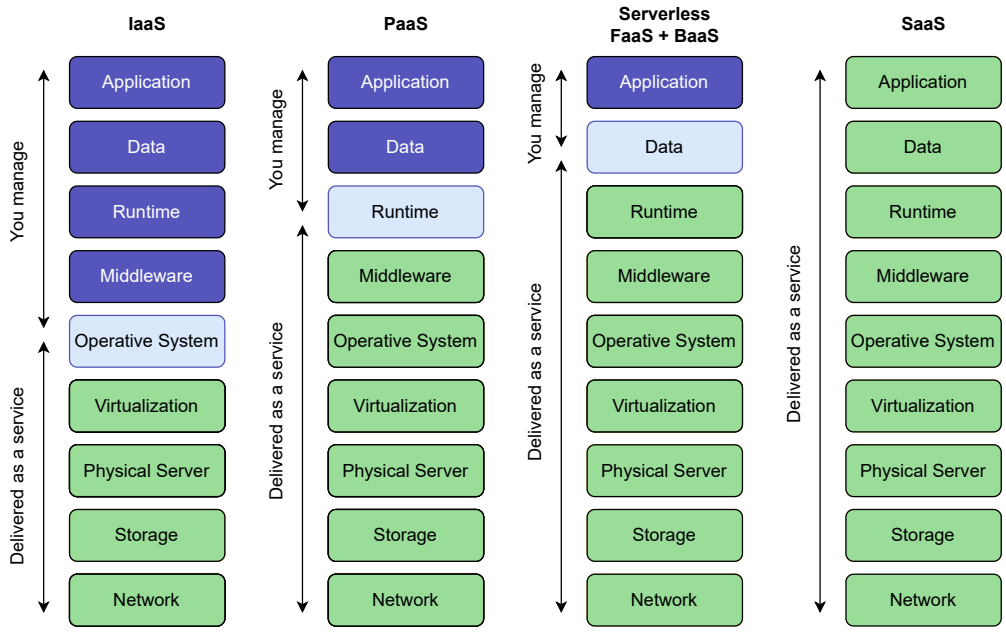


Figura 5.1: Comparación de los paradigmas más importantes utilizados en el entorno de la computación en la Nube.

+ Aplicación como suma de funciones: Esto permite actualizar, parchear, corregir o añadir nuevas funciones rápidamente a una aplicación. No es necesario realizar cambios en toda la aplicación, sino que los desarrolladores pueden actualizar cada función de forma independiente.

+ Automatización: El proceso no requiere la carga de protocolos de contingencia, ya que la tecnología ofrece tolerancia a los errores de la aplicación.

+ Funciones cercanas: La aplicación no se aloja en un servidor de origen. Las funciones se despliegan en la Nube cuando son necesarias. Esta tarea la realizará el proveedor. Las colocará lo más cerca posible del usuario para reducir la latencia.

+ Ahorro: se paga durante la ejecución del código.

– Pérdida de control sobre la infraestructura: El proveedor de servicios en la Nube controla la infraestructura subyacente, por lo que no podrás personalizarla u optimizarla para satisfacer tus necesidades específicas.

– Preocupaciones de seguridad: los desarrolladores de funciones deben

cuidar los datos críticos de la empresa, las filtraciones de secretos, la denegación de servicio, etc.

– Pruebas y depuración exigentes: La depuración es más complicada, porque los desarrolladores no tienen visibilidad de los procesos del *backend*, y porque la aplicación se divide en funciones separadas más pequeñas.

– No está diseñado para procesos a largo plazo: Los proveedores cobran por el tiempo que el código está en ejecución, puede costar más ejecutar una aplicación con procesos de larga duración en una infraestructura sin servidor que en una tradicional.

De acuerdo con las características de las arquitecturas *serverless*, éstas han evolucionado hasta convertirse en una herramienta muy útil para desplegar alguna funcionalidad en un entorno de producción pagando sólo cuando se ejecuta, sin invertir en la infraestructura (Taibi et al., 2021). Los sistemas de codificación de vídeo basados en HAS para un escenario de transmisión de vídeo bajo demanda podría ser un caso de uso adecuado para ser desplegado sobre una arquitectura *serverless*. En este tipo de escenario, el vídeo original se divide en varios segmentos (o chunks) con una duración de entre 2 y 4 segundos, y cada segmento suele codificarse para diferentes tasas de bits y/o diferentes resoluciones. Para cada una de estas codificaciones o conjunto de codificaciones por segmento, la Nube podría ejecutar un fragmento de código encapsulado en una función, realizando una asignación dinámica de recursos. La plataforma *serverless* sería capaz de ejecutar en paralelo la codificación de cada segmento ya que *serverless* permitiría escalar automáticamente el número de réplicas que ejecutan la función en caso de grandes necesidades, lo que podría reducir el tiempo total de codificación.

Con todos estos datos, cabe destacar la importancia de desarrollar sistemas de codificación basados en Nube con altas prestaciones, que sean altamente escalables y presenten un despliegue automatizado o cuasi-automatizado.

Por estos motivos, en este capítulo presentamos un sistema de codificación de vídeo basado en el paradigma de arquitectura *serverless* sobre una infraestructura de Nube. Nuestra propuesta se basa en la plataforma sin servidor de código abierto Knative (Knative, 2022) desplegada sobre una infraestructura con recursos limitados. El sistema implementado ha sido probado en varios escenarios con el fin de analizar su escalabilidad y rendimiento para codificar vídeos y calcular una métrica de calidad de referencia completa.

Las principales aportaciones de este trabajo son[1]:

- Desarrollo de funciones *serverless* de codificación de vídeo (x264 y AV1) y evaluación de la calidad de vídeo (VMAF) basadas en eventos, encapsuladas en contenedores ligeros.
- Análisis del comportamiento de las funciones desarrolladas en una plataforma *serverless* en local con recursos limitados. Se han realizado experimentos sobre escalabilidad y consumo de recursos.
- Análisis de la solución propuesta para vídeo bajo demanda para el *streaming* DASH con multiresolución, mostrando su rendimiento y comportamiento variando el número y tipo de réplicas.
- Desarrollo y análisis de *pipeline* de eventos para codificar segmentos de vídeo y obtener la calidad de vídeo codificado a partir de un único evento.

Este capítulo está organizado de la siguiente manera. En la Sección 2 se presentan nuestras funciones *serverless* de codificación de vídeo basadas en FaaS. La Sección 3 analiza la escalabilidad y el consumo de recursos de las funciones propuestas en un banco de pruebas real con vídeos 4K. Se realiza una comparación del comportamiento entre réplicas *fat* (con seis vCPUs) y *slim* (con una sola vCPU). Se evalúa el rendimiento de la propuesta para la codificación en multiresolución. Además, se presenta y analiza un *pipeline* para codificar segmentos y calcular la calidad del segmento codificado basándose en los eventos de la Nube. La sección 4 presenta una comparación cualitativa de nuestra propuesta con trabajos anteriores sobre codificación de vídeo sin servidor. Por último, en la Sección 6 se presentan las conclusiones.

5.2. Funciones serverless dirigidas por eventos para el procesamiento de flujos multimedia

En esta sección, se describen las principales características de la plataforma *serverless* Knative, utilizada en nuestro enfoque de diseño. Además,

[1]Se pueden obtener ficheros de despliegue de ejemplo utilizando git clone https://github.com/cloudmedialab-uv/serverless-video.git

describimos la arquitectura de la plataforma y sus principales componentes, así como la implementación de las funciones *serverless* basadas en eventos llevadas a cabo.

5.2.1. Plataforma serverless: Knative

Existen algunas plataformas *serverless* de código abierto: Knative, Nuclio, OpenFaaS, o Kubeless. Las características que ofrecen estas plataformas han sido analizadas en (Li et al., 2021). En nuestro despliegue, hemos seleccionado Knative porque el escalado de réplicas se puede controlar teniendo en cuenta la concurrencia (esto es importante en nuestra aplicación ya que sólo permitimos una ejecución concurrente por réplica ya que el trabajo realizado en cada réplica consume todos los recursos asignados). Además, ofrece arranque en frío/caliente, escalado a cero, y tiene soporte para CloudEvents (CloudEvents, 2022).

CloudEvents es una especificación para describir datos de eventos que simplifica la declaración y entrega de eventos a través de servicios y plataformas. Esta especificación depende de la Cloud Native Computing Foundation y ha sido adoptada por los principales proveedores de Nube y empresas de SaaS. La especificación describe cómo encapsular CloudEvents en diferentes protocolos como AMQP, HTTP, Kafka, MQTT, websockets, Protobuf, etc. En nuestra aplicación, los CloudEvents se encapsularán en mensajes HTTP. Por lo tanto, los metadatos CloudEvent se añadirán como cabeceras HTTP mientras que los datos del evento se encapsularán en formato JSON en el cuerpo del mensaje HTTP.

Cuando se despliega un recurso de tipo *Knative service*, el pod creado contiene el contenedor de usuario (donde se ha encapsulado la función) y el contenedor *queue-proxy* (patrón sidecar) que expone métricas y es utilizado por Knative para controlar la concurrencia, tiempos de espera, reintentos, etc., tal y como se muestra en la Figura 5.2.

Knative soporta el uso de CloudEvents para desplegar funciones basadas en eventos. El usuario debe crear un *broker* que reciba los eventos de una fuente de eventos. El *broker* puede tener varios *triggers* que utilizan los atributos CloudEvent `type` y/o `source` para determinar qué consumidor debe recibir el CloudEvent como se muestra en la Figura 5.3.

El *sink* o consumidor del evento puede ser un *deployment* (un recurso

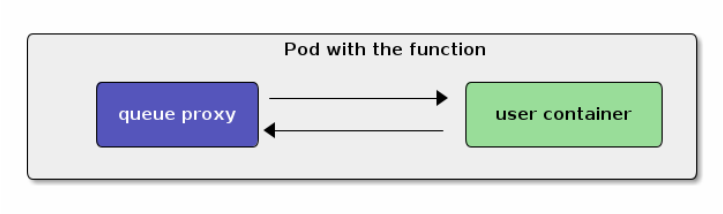


Figura 5.2: El patrón sidecar: el pod con la función contiene un contenedor adicional. Este contenedor envía las solicitudes, obtiene la respuesta, realiza comprobaciones de estado y reintentos, expone métricas, etc.

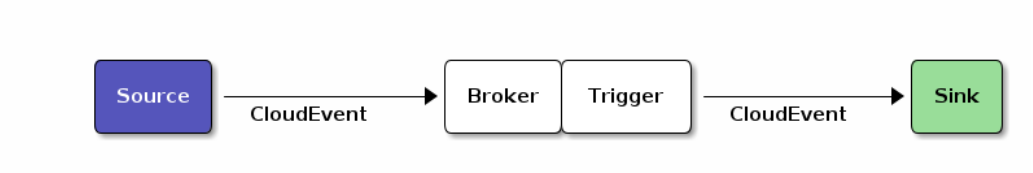


Figura 5.3: Una fuente envía un CloudEvent al *broker*. El *trigger* envía el evento a un receptor (o *sink*) según el tipo y/o los atributos de origen del evento.

estándar de Kubernetes) o un servicio Knative (*serving*). En este último caso, la plataforma *serverless* proporciona escalabilidad automática, control de concurrencia, etc.

Al instalar Knative, se debe instalar una capa de red y algunas posibilidades son Kourier, Istio, o Contour. Nosotros hemos optado por Contour ya que lee las propiedades de Knative que controlan los tiempos de espera de las funciones. Estas propiedades se establecen en el *configmap* `config-defaults` en el espacio de nombres `knative-serving`. En nuestro caso, hemos modificado los valores por defecto para permitir la ejecución de funciones largas como se muestra en Listing 2.

La Figura 5.4 muestra la infraestructura desplegada. Knative se ha instalado en un clúster de 9 máquinas virtuales con Kubernetes y se utiliza para desplegar diferentes funciones `serverless` basadas en eventos. Los vídeos se dividen en segmentos y se almacenan en un servidor *download* (en este caso, un contenedor Nginx que se ejecuta en un servidor).

Se ha utilizado un registro privado que almacena las imágenes de los diferentes contenedores. Todas las imágenes se han etiquetado con este registro, y Kubernetes se ha configurado con las credenciales para obtener imágenes

```
apiVersion:  v1
kind:  ConfigMap
metadata:
  name:  config-defaults
  namespace:  knative-serving
data:
  revision-timeout-seconds: "1000"
  max-revision-timeout-seconds: "2000"
```

Listing 2: Propiedades de tiempo de espera modificadas que se utilizan para determinar los tiempos de espera de la función.

de este registro privado.

Cuando se procesa un segmento de vídeo en una función, el resultado se sube a un servidor (ya que las funciones son efímeras) que está ejecutando un servicio de *upload* (desarrollado en Tomcat).

Por otra parte, cuando un usuario desea codificar un vídeo, debe generarse un CloudEvent para codificar cada segmento. Los eventos se envían a la capa de red *gateway* que se encarga de entregarlo a una réplica que ejecuta la función correspondiente.

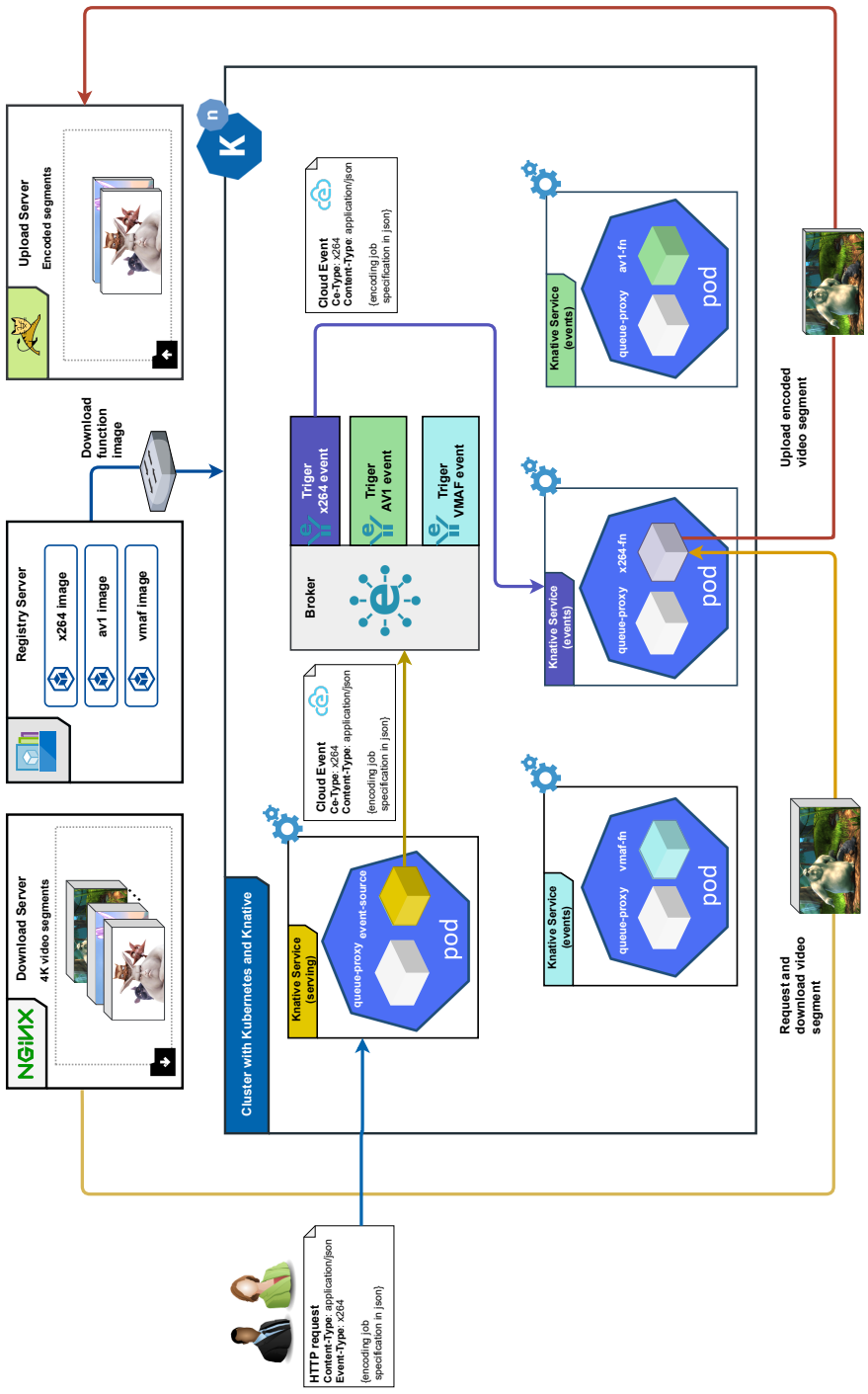


Figura 5.4: Un clúster con Kubernetes y Knative donde se despliegan las funciones. Un servidor de descarga con los vídeos, un servidor de carga para subir los resultados y un registro de imágenes de contenedores están conectados al clúster.

5.2.2. Implementación de funciones

La plataforma *serverless* Knative no impone ninguna restricción sobre la implementación de las funciones. En nuestro caso, optamos por hacer el desarrollo utilizando Java debido a nuestros conocimientos por proyectos anteriores. Usando este lenguaje, la función se puede desarrollar usando Tomcat, Quarkus, Spring, Micronaut, etc. para exponer un endpoint HTTP.

En este trabajo, las funciones se han implementado utilizando un Tomcat embebido modificado[2] que proporciona abstracciones para trabajar con CloudEvents (Gutiérrez-Aguado, 2022). El uso de este entorno de ejecución conduce a imágenes más pequeñas ya que Tomcat tiene menos dependencias en comparación con los otros entornos de ejecución de aplicaciones. El Tomcat embebido modificado añade una clase abstracta `CloudEventListener`, que extiende `GenericServlet`, con el método abstracto `consumeEvent(...)` que será llamado cuando se reciba un HTTP POST. El primer argumento de este método es el `CloudEvent`, el segundo es el `HttpServletRequest`, y el tercero es de tipo `HttpServletResponse`.

Utilizando este servidor de aplicaciones, hemos implementado tres funciones *serverless* dirigidas por eventos:

- Una función para codificar un vídeo usando x264 (Guo et al., 2018).
- Una función para codificar un vídeo utilizando AV1 (Guo et al., 2018).
- Una función para calcular el VMAF (Li et al., 2016b) entre un vídeo de referencia y un vídeo distorsionado.

Los esquemas de muestra de las funciones implementadas se muestran en los listados 3 y 4.

Estas funciones se ejecutarán en un entorno *serverless*, donde las réplicas son efímeras y pueden escalarse a cero. Por lo tanto, para cada invocación, el trozo de vídeo debe ser descargado, procesado (construyendo una llamada a `ffmpeg`), y finalmente subido a un servicio de carga externo.

Una instancia de la `FunctionX264` puede ser registrada como un `Servlet` en una instancia de `Tomcat` para procesar peticiones HTTP.

[2]El proyecto es un fork de Tomcat y puede ser clonado desde `https://github.com/jgutie2r/tomcat`

```
class FunctionX264 extends CloudEventListener{
   public void consumeEvent(CloudEvent ev,
                            HttpServletRequest req,
                            HttpServletResponse resp){
      // Deserializar el cuerpo del CloudEvent desde JSON
      // Descargar el video
      // Codificar el video
      // Subir el video codificado
      // Si se solicita un procesamiento adicional, envía un Cloud event
   }
}
```

Listing 3: Esquema de la función que codifica un trozo de vídeo usando x264.

```
class FunctionVMAF extends CloudEventListener{
   public void consumeEvent(CloudEvent ev,
                            HttpServletRequest req,
                            HttpServletResponse resp){
      // Deserializar el cuerpo del CloudEvent desde JSON
      // Descarga los dos videos (referencia y distorsionado)
      // Calcular VMAF
      // Subir resultados .json de VMAF
   }
}
```

Listing 4: Esquema de la función que calcula la métrica de calidad VMAF de un segmento de vídeo codificado

Una vez desplegadas estas funciones, pueden activarse con peticiones HTTP. Listing 5 muestra un ejemplo de llamada enviada al *broker* a través de *Envoy* (que proporciona un proxy inverso L4/L7 de alto rendimiento) expuesto en el puerto 8080.

Como estas funciones hacen un uso intensivo de la CPU, como se verá en la sección 5.3.3, no permitimos la concurrencia, es decir, sólo se envía un CloudEvent a cada réplica de las funciones —ya que, se pretende maximizar el rendimiento de cada réplica—. Cuando la réplica termina el trabajo de codificación, está lista para recibir y procesar el siguiente evento o detenerse si no llegan nuevas peticiones.

```
curl http://localhost:8080/video-coding/video-coding-broker \
  -H 'Ce-Type: x264' \
  -H 'Ce-Source: /curl' \
  -H 'Ce-specversion: 1.0' \
  -H 'Ce-Id: 1' \
  -H 'Host: broker-ingress.knative-eventing.svc.cluster.local'\
  -d '{
      "sourceUrl":"https://download-server/video/video.mp4",
      "crf":"40",
      "frameRate":"23.98",
      "extraParams":"-threads 6",
      "videoNameDestination":"video-crf-40.mp4",
      "destUrl":"http://upload-server/upload",
      "bucketParamName":"bucket",
      "bucketParamValue":"encodedVideos/",
      }'
```

Listing 5: Ejemplo de solicitud HTTP para codificar un segmento. Como el atributo del evento de Nube `Ce-Type` es `x264`, el activador enviará este evento a la función x264.

5.3. Experimentos

En esta sección describimos el montaje experimental y las condiciones utilizadas para validar la arquitectura propuesta. Hemos llevado a cabo experimentos para validar la escalabilidad y el buen comportamiento en términos de rendimiento y consumo de recursos (CPU y memoria) de las funciones de codificación de vídeo (x264 y AV1) y de la métrica de calidad de vídeo (VMAF). Además, hemos evaluado el uso de *pipelines* de procesamiento dirigidos por eventos.

5.3.1. Entorno de pruebas

La infraestructura utilizada para desplegar los componentes propuestos, que se muestra en la Figura 5.4, consta de cinco máquinas físicas conectadas a través de un switch *Gigabit Ethernet.*Una máquina (Intel(R) Xeon(R) CPU E3-1220 v6 @3.00GHz, 4 cores, 16 GB RAM) ejecuta, entre otros servicios, un registro privado en contenedores con todas las imágenes de las funciones desarrolladas.

Otra máquina (2 Intel(R) Xeon(R) Silver 4210 CPU @2.10Ghz con 10 núcleos cada una y Hyper-threading, 128 GB RAM) ofrece un servicio de descarga containerizado (Nginx) donde se almacenan todos los segmentos. Esa máquina, ejecuta dos VMs no relacionadas con este experimento que consumen 30 vCPUs y 96 GB RAM.

La tercera máquina, es un servidor dedicado (Pentium(R) Dual-Core CPU E6600 @3.06GHz, 2 cores, 4 GB RAM), que ejecuta un servicio de subida containerizado (desarrollado usando Tomcat) para almacenar los vídeos codificados y los resultados del cálculo de la métrica de calidad VMAF.

Los otros dos servidores, con 104 vCPUs disponibles, tienen las siguientes características:

- Server1: 2 Intel Xeon Silver 4216 CPU @ 2.10GHz with 16 cores each y Hyper-threading (up to 64 vCPUs), 128 GB RAM @2666 MHz.
- Server2: 1 Intel Xeon Gold 5218R CPU @ 2.10GHz with 20 cores y Hyper-threading (up to 40 vCPUs), 64 GB RAM @3200 MHz.

Estos ejecutan un clúster de 9 máquinas virtuales con Ubuntu 20.04, Kubernetes (versión 1.23.4) y Knative (versión 1.4.0). El clúster Kubernetes consta de un nodo maestro y 8 workers. El nodo maestro tiene 10 vCPUs y 16 GB RAM, mientras que los workers tienen 6 vCPUs y 10 GB RAM. El nodo maestro y del Worker1 al Worker4 se ejecutan en el `Servidor1`, mientras que del Worker5 al Worker8 se ejecutan en el `Servidor2`.

Cada VM dispone de una imagen de disco almacenada en un SSD Crucial dedicado de 240 GB. Estas VMs fueron creadas con `virt-install` y configuradas con Ansible.

Como podemos ver, la plataforma *serverless* está instalada en un entorno heterogéneo con características de hardware ligeramente diferentes. Esto no afecta a la funcionalidad general de la aplicación, pero como comentaremos en la subsección 5.3.3.1, tiene un impacto en el speedup en comparación con un sistema homogéneo.

Los vídeos 4K utilizados en los experimentos se enumeran en la Tabla 5.1. Los vídeos se han descargado, transcodificado utilizando x264 a alta calidad (CRF=10, que genera vídeos codificados sin pérdidas visuales), y finalmente dividido para obtener trozos temporales de vídeo de 2 segundos (Lederer,

Tabla 5.1: Vídeos 4K utilizados para realizar los experimentos.

Vídeo	**Duración (s)**	**segmentos**	**Tamaño(GB)**
Ancient Thought (ANC)	188	94	2.9
Eldorado (ELD)	184	92	4.7
Indoor Socker (IND)	252	126	8.0
Skateboarding (SKA)	276	138	7.7

2020). Los tamaños mostrados en la Tabla 5.1 son el tamaño final tras la transcodificación y la división.

Los códecs encapsulados en las funciones son H.264 (x264) y AV1 (rav1e) integrados en `FFmpeg` (FFmpeg, 2022). Las imágenes del contenedor de las funciones utilizan una imagen base de Java 11, y se utilizó un patrón multi-etapa para copiar el binario de FFmpeg (con soporte para múltiples codecs) a la imagen. La Tabla 5.2 muestra los tamaños finales de las imágenes de los contenedores que encapsulan las diferentes funciones.

Tabla 5.2: Tamaño de las imágenes contenedoras de las funciones desarrolladas.

Función	**Tamaño de la imagen de contenedor (MB)**
x264-fn	126
av1-fn	126
vmaf-fn	144

En todos los experimentos, no se ejecutó ninguna otra función en la infraestructura.

5.3.2. Impacto de la invocación sin réplicas en ejecución frente a la invocación con réplicas en ejecución

En esta subsección, se analiza el impacto de la invocación de la función cuando no hay réplicas de la función en ejecución (o *cold start*) en nuestro despliegue. Para este experimento, invocamos la función `x264-fn` utilizando ocho réplicas *fat*.

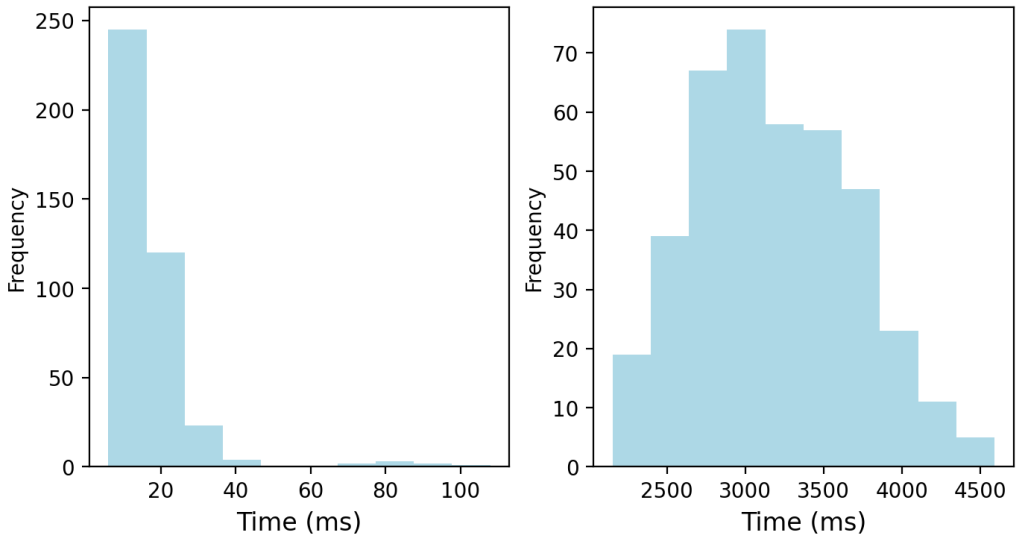


Figura 5.5: Distribución del tiempo transcurrido desde la creación del evento hasta la recepción del evento en la función. Izquierda: arranque en caliente. Derecha: arranque en frío.

Cada réplica está configurada para consumir 6 vCPUs, por lo que cada réplica solicita todas las vCPUs disponibles del nodo worker donde se ha ejecutado. Se consideran dos escenarios:

- No hay réplicas en ejecución (arranque en frío). En este caso, Knative debe iniciar todas las réplicas para hacer frente a los eventos entrantes.
- Ocho réplicas en ejecución (arranque en caliente o *warm start*).

En ambos casos, realizamos ocho invocaciones simultáneas para codificar segmentos de vídeo y medimos el tiempo transcurrido desde la creación del evento hasta el momento en que éste se recibe en la función. Esto se ha repetido 50 veces (400 invocaciones en cada caso) y en el escenario de arranque en frío, nos aseguramos de que entre cada iteración todas las funciones han finalizado y han escalado a cero. En la Figura 5.5 se muestran los histogramas de la distribución de tiempos tanto en arranque en caliente como en arranque en frío. El tiempo medio en arranque en caliente es de 17.7 ms mientras que en el caso de arranque en frío el tiempo medio es de 3179.4 ms.

```
# Llamada de muestra para codificar un segmento usando x264 (llamada realizada dentro de
↪  la función)
ffmpeg -i segmento1.mp4 -c:v libx264 -threads 6 -crf 23 output.mp4

# Llamada de muestra para codificar un segmento usando AV1 (llamada realizada dentro de
↪  la función)
ffmpeg -i segmento1.mp4 -c:v rav1e -crf 35 -row-fmt 1 -cpu-used 4 \
   -threads 6 -tile-col 4 -rav1e speed=10 output.mp4

# Ejemplo de llamada para obtener el VMAF de un segmento codificado en 4K (llamada
↪  realizada dentro de la función)

ffmpeg -y -hide_banner -r 23.98 -i reference.mp4 -r 23.98 -i distorted -lavfi
"[0:v]setpts=PTS-STARTPTS[reference];[1:v]setpts=PTS-STARTPTS[distorted];
[distorted][reference]libvmaf=log_fmt=json:log_path=/tmp/segmento1_23_2160p_vmaf.json:
model_path=/usr/local/share/model/vmaf_4k_v0.6.1.json:n_threads=6" -f null -
```

Listing 6: Ejemplo de llamadas dentro de las funciones para codificar un segmento usando x264 y AV1. La última llamada de muestra es para obtener el VMAF de un segmento codificado.

5.3.3. Escalabilidad y rendimiento de las funciones

En esta subsección, hemos realizado varios experimentos para mostrar la escalabilidad y el rendimiento de la infraestructura propuesta. Para la escalabilidad, evaluamos el rendimiento de las funciones de codificación `x264-fn` y `av1-fn`, de 1 a 8 réplicas que solicitan 6 vCPUs cada una, denominadas réplicas *fat*. Estos resultados se comparan con el uso de réplicas que solicitan 1 vCPU cada una, denominadas réplicas *slim*, cuando ejecutan la función `x264-fn`. Con esta prueba, podemos ver el comportamiento de la propuesta para distintos tipos de granularidad.

Por último, se realiza un análisis de rendimiento detallado del consumo de memoria por réplica y del uso de CPU por trabajador de las funciones de codificación y calidad para el caso de 8 réplicas *fat*.

5.3.3.1. Escalabilidad con réplicas fat

En este experimento se ha medido la escalabilidad y el rendimiento de las funciones de codificación. En todas las pruebas se ha utilizado el arranque en frío y se ha variado el número máximo de réplicas permitidas de 1 a 8: $\{1, 2, 4, 8\}$. Todos los trozos del vídeo ANC se codificarán utilizando las funciones `x264-fn` y `av1-fn`.

Tabla 5.3: Tiempos promedio por segmento (en milisegundos) para descargar, codificar y subir los 94 segmentos en resolución 4K de ANC a medida que aumenta el número de réplicas *fat* x264. La última columna muestra el tiempo total (en segundos) para completar la codificación de todos los segmentos.

	Tiempos promedio por segmento para réplicas fat x264			
Réplicas	***Descargar (ms)***	***Codificar (ms)***	***Subir (ms)***	***Total (s)***
1	285.4	6020.4	172.2	617.4
2	299.6	6434.5	177.4	333.4
4	302.4	7203.0	177.6	191.6
8	449.5	7867.3	230.4	112.6
	Tiempos promedio por segmento para réplicas fat AV1			
Réplicas	***Descargar (ms)***	***Codificar (ms)***	***Subir (ms)***	***Total (s)***
1	285.6	34102.7	39.9	3256.1
2	290.8	35453.5	40.8	1698.9
4	306.5	40203.5	42.7	980.5
8	373.0	44520.0	47.0	547.3

La Tabla 5.3 muestra los tiempos de codificación de los 94 trozos del vídeo ANC 4K a medida que aumenta el número máximo de réplicas de la función. En todos los casos, los 94 eventos se envían en paralelo. En el caso de x264, el tiempo necesario para que una réplica codifique todos los segmentos es de 617.4 segundos, mientras que el tiempo para codificar todos los segmentos del vídeo utilizando 8 réplicas es de 112.6 segundos (el aumento de velocidad en comparación con una réplica es de 5.5 y la eficiencia es del 68 %). En el caso de AV1, el tiempo para codificar todos los trozos del vídeo con una réplica es de 3256.1 segundos y este tiempo se reduce a 547.3 segundos con 8 réplicas (el aumento de velocidad en este caso es de 5.9 y la eficiencia es del 74 %).

En este experimento, el planificador ha asignado, para ambos códecs, las cuatro primeras réplicas en Worker5 a Worker8 (en Server2 con características de hardware ligeramente mejores). Por tanto, el caso base (el tiempo con una réplica) se ha ejecutado en una VM en el Servidor2.

En el experimento con 8 réplicas, cuatro se ejecutaron en Worker5 a Worker8 (Servidor 2) y las otras cuatro en Worker1 a Worker4 (en Servidor1 con características de hardware ligeramente inferiores). El aumento de velocidad habría sido mayor en un sistema homogéneo con las características del Servidor2.

La Figura 5.6 muestra la distribución de trabajos por réplica a medida que aumenta el número de réplicas utilizando el codificador x264 y AV1, respectivamente. Por un lado, los trabajos tienen una duración diferente en función de la complejidad espacial y temporal del segmento de vídeo. Por otro lado, el tiempo de codificación de cada segmento es diferente para cada codificador debido a su funcionamiento intrínseco. Por lo tanto, el número de trabajos de cada réplica depende del codificador y de la complejidad de vídeo del segmento. Pero, en todos los casos, podemos ver que el sistema propuesto intenta equilibrar los trabajos entre las réplicas disponibles. Además, algunas réplicas pueden completar más trabajos que otras porque la entrega de eventos está controlada dinámicamente por los componentes de eventos de Knative.

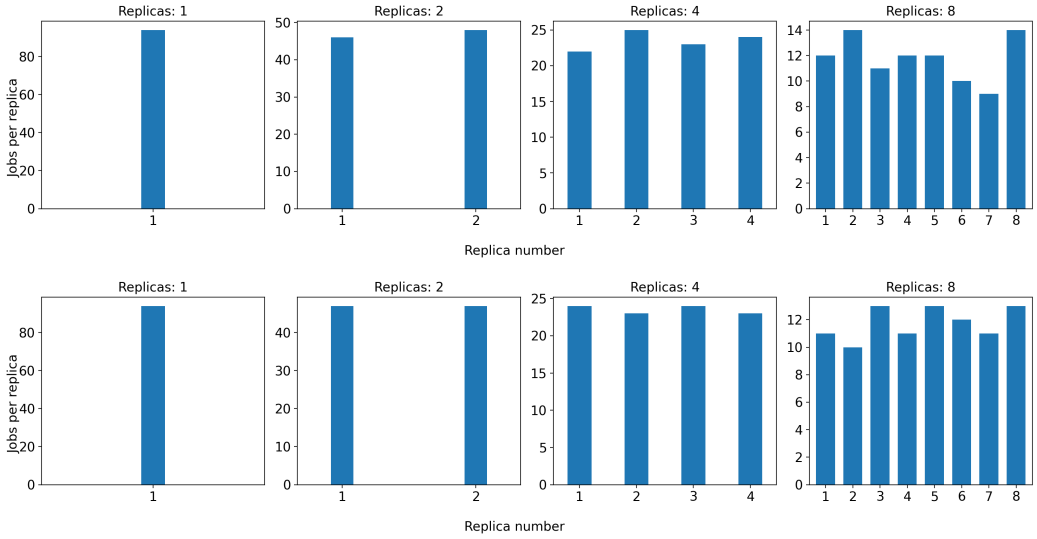


Figura 5.6: Número de trabajos ejecutados por réplica para codificar 94 segmentos de vídeo ANC utilizando réplicas fat x264 (arriba) y réplicas fat AV1 (abajo).

5.3.3.2. Escalabilidad con réplicas slim

En el último experimento de esta sección, probamos el comportamiento utilizando *slim replicas* que solicitan menos recursos computacionales. En este experimento, cada función solicita una vCPU y, dentro de la función, se lanza `ffmpeg` con un hilo. En este experimento el número máximo de `x264-fn` réplicas de función permitidas se incrementó a 48, consumiendo las mismas vCPUs que en el experimento anterior.

Tabla 5.4: Tiempos promedio por segmento (en milisegundos) para descargar, codificar y subir los 94 segmentos en resolución 4K de ANC utilizando 48 réplicas slim x264 (1 vCPU por réplica). La última columna muestra el tiempo total (en segundos) para completar la codificación de todos los segmentos del vídeo.

	Tiempos promedio por segmento para réplicas slim x264			
Réplicas	***Descargar (ms)***	***Codificar (ms)***	***Subir (ms)***	***Total (s)***
48	1973.9	28312.4	262.5	92.6

A través de este experimento queremos saber qué tipo de despliegue es más eficiente: uno con muchas réplicas con baja capacidad de procesamiento, o uno con pocas réplicas con alta capacidad de procesamiento.

Comparando la Tabla 5.4 con la Tabla 5.3, podemos ver que el tiempo total para codificar los 94 segmentos es de 92.6 segundos, un 18 % menos que los 112.6 segundos con el uso de 8 réplicas *fat* (usando 6 vCPU y 6 *threads*). El tiempo medio para descargar los segmentos ha aumentado porque ahora hay más funciones descargando simultáneamente los datos (las funciones tienen *vecinos ruidosos*). El tiempo medio para codificar cada segmento aumenta a 28.3 segundos, ya que ahora sólo hay un hilo para `ffmpeg`. Por último, en cuanto al número de trabajos computados por réplica, 5 réplicas han codificado 1 segmento, 40 réplicas han codificado 2 segmentos, y 3 réplicas han codificado 3 segmentos. Sumando todos los trabajos ($5\times1+40\times2+3\times3=94$), obtenemos todos los segmentos de la película analizada.

Para concluir esta subsección, se podría indicar que para vídeo streaming en directo, donde los segmentos deben codificarse lo más rápido posible para ser entregados, una configuración con réplicas *fat* utilizando todos los recursos disponibles es más apropiada ya que cada segmento se codifica más rápido comparado con el uso de réplicas *slim*. Sin embargo, para el vídeo bajo demanda pueden utilizarse réplicas *slim*, ya que se reduce el tiempo total necesario para completar la codificación.

5.3.3.3. Consumo de memoria para funciones de codificación de vídeo

En este experimento se obtuvo un consumo de memoria mediante réplicas *fat* al codificar los 94 segmentos de ANC con las funciones `x264-fn` y `av1-fn`.

El número máximo de réplicas para cada función se limitó a 8.

Como puede verse en la Figura 5.7 a), las réplicas que realizan la codificación x264 muestran un pico de consumo de RAM de 2.2 GB. La Figura 5.7 b) muestra el consumo de memoria por réplica utilizando la función `av1-fn`. En este caso, la función `av1-fn` que realiza la codificación muestra un pico de consumo de RAM de 5 GB. Esta función duplica la RAM necesaria para codificar con la función x264. Esto se debe a la forma en que funcionan los códecs. x264 es un códec de segunda generación ampliamente utilizado y muy eficiente debido a sus 20 años de trabajo continuo. Pero el tamaño máximo de macrobloque para x264 es 16x16, mientras que para AV1 es 128x128. Los macrobloques más grandes necesitan almacenar más información en la RAM para realizar tareas como la estimación y compensación del movimiento, la transformación, la cuantización, etc. (Kerdranvat et al., 2020).

Por tanto, el codificador determina los recursos que necesita la función y también el número máximo de réplicas que pueden ejecutarse simultáneamente en una infraestructura con recursos limitados.

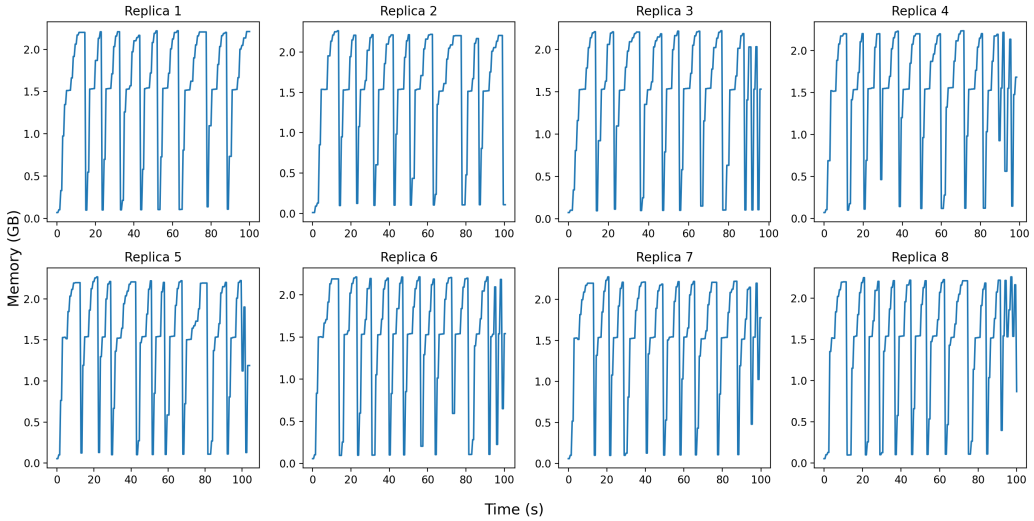
5.3.3.4. Consumo de CPU para funciones de codificación de vídeo

En este experimento, el consumo de CPU utilizando réplicas *fat* se obtuvo al codificar los mismos trozos de la subsección 5.3.3.3 con las funciones `x264-fn` y `av1-fn`. El número máximo de réplicas de la función se limitó a 8.

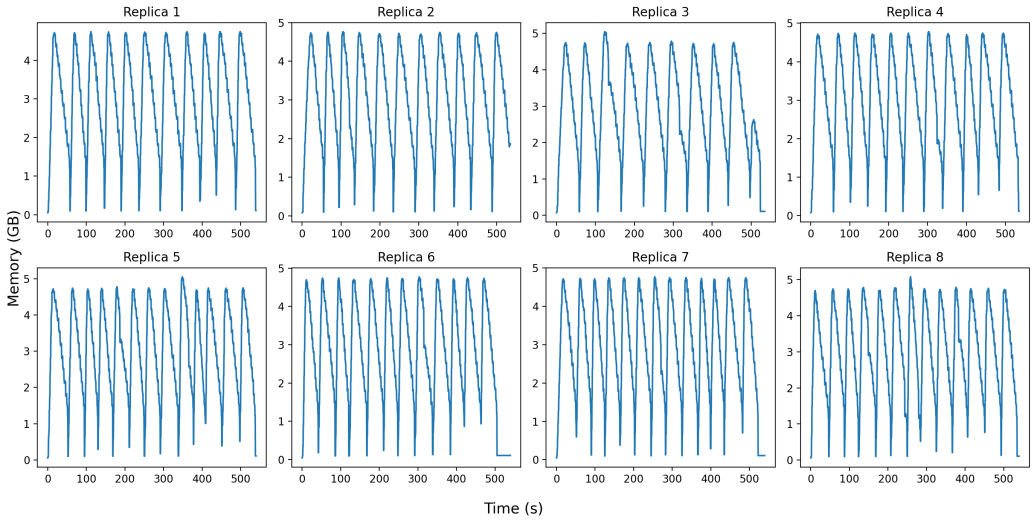
La Figura 5.8 muestra los boxplots del consumo de CPU (en %) por vCPU para cada nodo worker (el planificador ha asignado una réplica por worker) cuando están utilizando las funciones `x264-fn` y `av1-fn` respectivamente. Como puede verse en Listing 6, la llamada `ffmpeg` utiliza el mismo número de *threads* que vCPU tiene el nodo worker.

La Figura 5.8 a) muestra como la función `x264-fn` tiene todas las vCPU de cada réplica trabajando entre 60 % y 80 %, llegando a 100 % en muchas ocasiones. Este comportamiento es un poco diferente para la función `av1-fn` (ver Figura 5.8 b). Para este caso, las vCPU de cada réplica consumen un 50 % de media, llegando al 80 % en muchas ocasiones.

Las diferencias en el uso de CPU para las tareas de codificación se deben principalmente a la implementación del códec. También hay que tener en cuenta que el codificador x264 lleva más de 20 años mejorándose, mientras



a)



b)

Figura 5.7: Consumo de memoria (en GB) frente a tiempo (en segundos) por réplica con un máximo de 8 réplicas fat de las funciones de codificación: a) `x264-fn`, b) `av1-fn`.

que el códec AV1 sólo existe desde hace 3 años.

5.3.3.5. Consumo de memoria para funciones de calidad de vídeo

En este experimento, se mide el consumo de recursos de las réplicas *fat* VMAF para segmentos previamente codificados utilizando x264 y AV1. Cuando se descarga el segmento codificado, hay que descodificarlo para realizar una comparación con el vídeo original de alta calidad.

La Figura 5.9 muestra el consumo de memoria por réplica (una por nodo) durante el experimento utilizando 8 réplicas *fat* de la función `vmaf-fn`. Para ambos códecs podemos ver que la función que ejecuta VMAF muestra un pico de consumo de 2.85 GB de RAM para los segmentos.

Para obtener un valor VMAF, la función descodifica los fotogramas del vídeo de referencia y descodifica los fotogramas del vídeo distorsionado para obtener una medida a nivel de fotograma. Estos resultados se combinan para obtener la puntuación de calidad final.

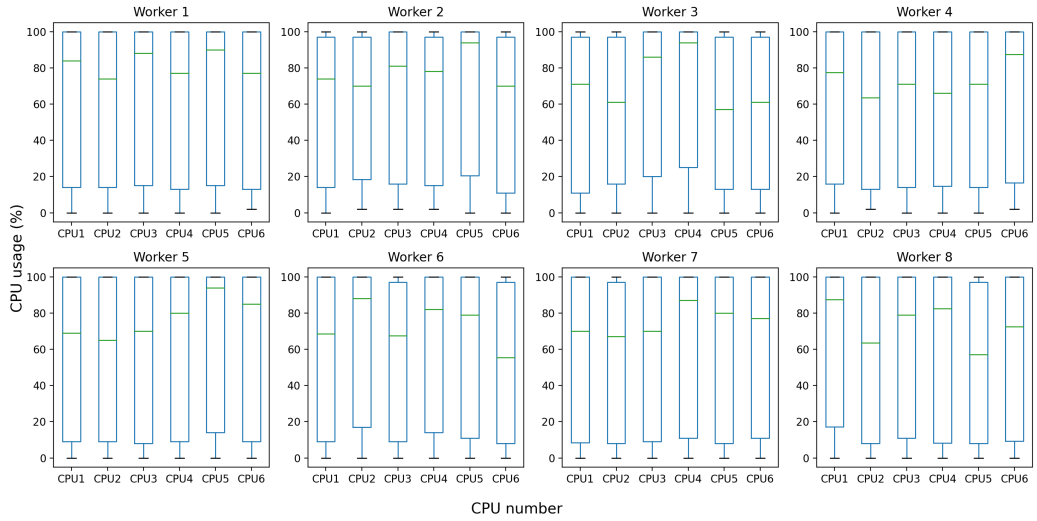
La Figura 5.9 muestra que esta función `vmaf-fn` tiene un comportamiento similar, independientemente del descodificador utilizado en los vídeos en términos de RAM requerida. Por tanto, la RAM necesaria para calcular VMAF será independiente del decodificador pero dependerá de la resolución. Para una resolución FHD (1920x1080), que tiene menos píxeles que la resolución 4K (3840x2160), la memoria consumida será menor.

5.3.3.6. Consumo de CPU para funciones de calidad de vídeo

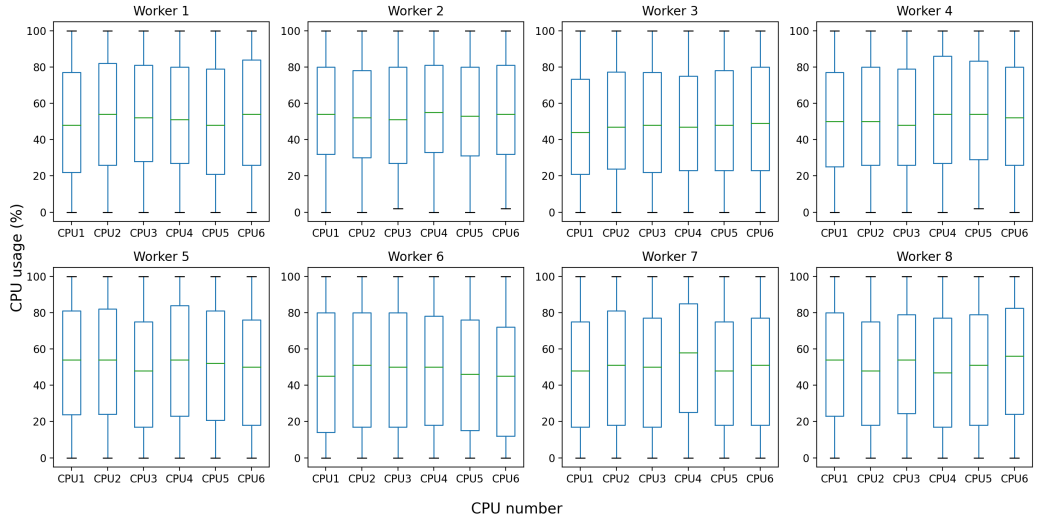
La Figura 5.10 muestra los boxplots del consumo de CPU (en %) por número de CPU para cada nodo worker con un máximo de 8 réplicas *fat* de la función `vmaf-fn` cuando recibe vídeos x264 y AV1.

Podemos ver un comportamiento similar al observado en la Figura 5.9, donde no hay dependencia relevante del decodificador del vídeo de entrada. Las réplicas en todos los workers consumen cerca del 100 % de las vCPUs para decodificar y calcular la puntuación VMAF.

En resumen, cuando los vídeos se codifican utilizando x264, los resultados en términos de consumo de RAM y uso de CPU en la función VMAF son muy similares a los obtenidos utilizando AV1. Por tanto, el consumo de recursos,

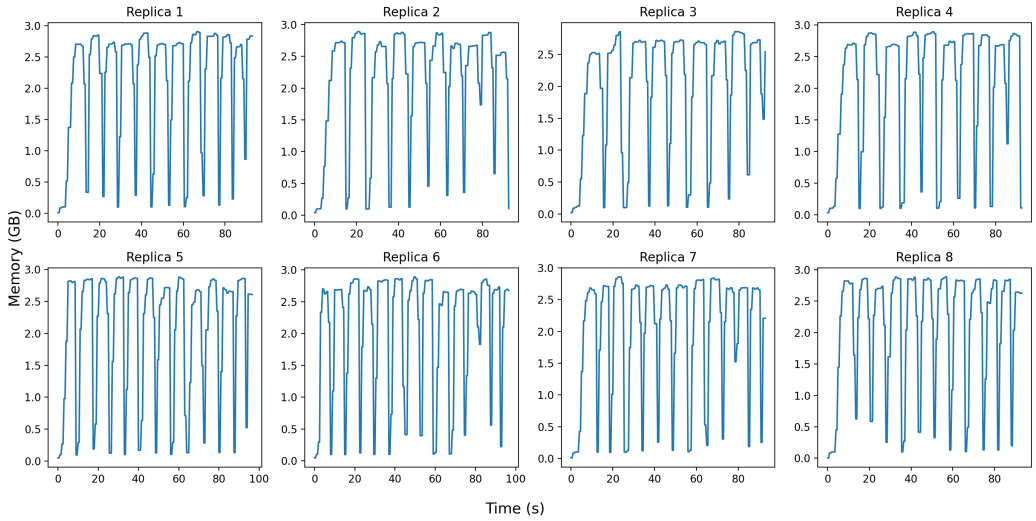


a)

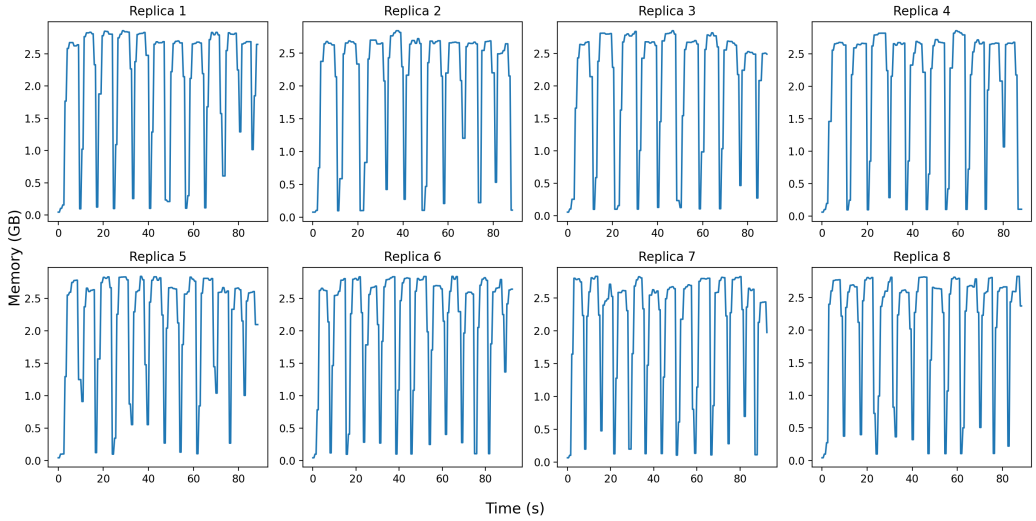


b)

Figura 5.8: Consumo de CPU (en %) por número de CPU para cada nodo worker con un máximo de 8 fat réplicas de las funciones de codificación: a) `x264-fn`, b) `av1-fn`.



a)



b)

Figura 5.9: Consumo de memoria (en GB) vs tiempo (en segundos) por contenedor con un máximo de 8 réplicas fat de la función `vmaf-fn` cuando los segmentos han sido codificados usando: a) x264, b) AV1.

en este caso, no depende del decodificador (x264 o AV1).

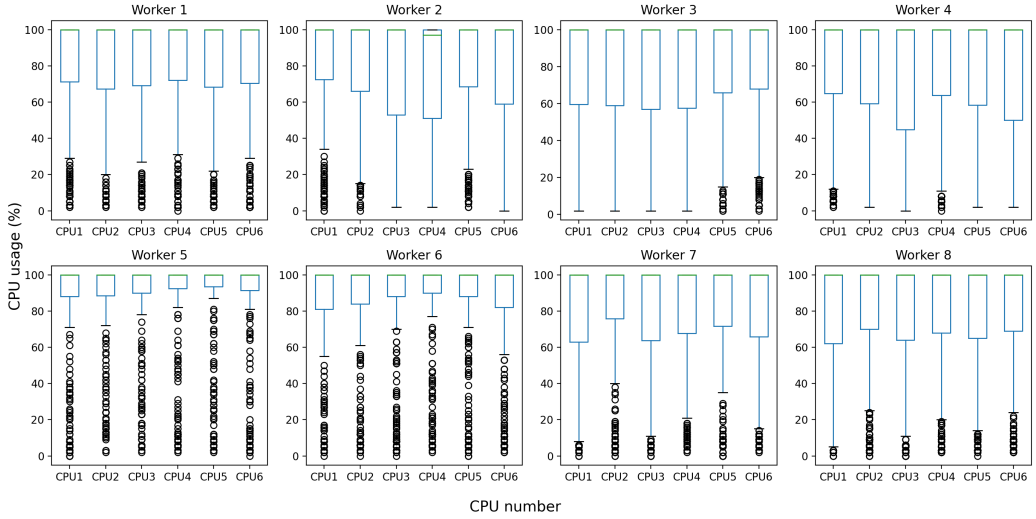
5.3.4. Codificación de vídeo multiresolución

En DASH, un segmento se codifica en múltiples representaciones, por ejemplo, diferentes resoluciones y tasas de bits. El cliente solicitará la mejor representación en función de las condiciones cambiantes de la red (Seufert et al., 2014). En este experimento, generamos múltiples representaciones de cada segmento de vídeo para cada invocación de la función.

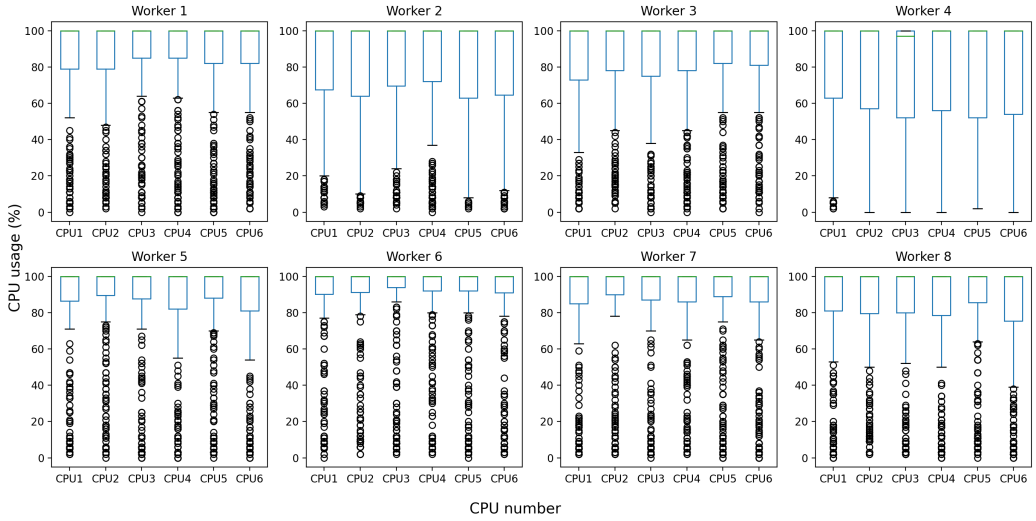
Se han codificado cuatro vídeos (ANC, ELD, IND y SKA) generando múltiples representaciones para cada segmento en una única llamada. Cada invocación a la función descarga el segmento asignado y codifica el vídeo en 4 representaciones con diferentes resoluciones y *bitrates*, luego todas las representaciones generadas se cargan utilizando una única petición POST `multipart/form-data`. En estos experimentos, utilizamos el arranque en frío y el número máximo de réplicas permitidas fue de 8.

Listing 7 y Listing 8 muestran llamadas de ejemplo utilizadas dentro de las funciones `x264-fn` y `av1-fn`, respectivamente, para codificar un segmento de vídeo 4K generando cuatro resoluciones con diferentes tasas de bits. Las tasas de bits por resolución utilizadas para x264 son las recomendadas por (IBM Watson Media Support Center) y para AV1 en resoluciones 480p, 720p, y 1080p las tasas de bits son las recomendadas en el trabajo de (Guo et al., 2018) y para resoluciones 2160p o 4K, las tasas de bits fueron tomadas de (Li et al., 2019). La adaptación de la llamada `FFmpeg` se realiza en el cuerpo de la petición `curl` utilizando los `extraEncoderParams` que se muestran en Listing 5.

Comparando los tiempos de codificación en la Tabla 5.5, podemos ver que los tiempos promedio de subida para AV1 son menores que los tiempos promedio de subida para x264, ya que el tamaño de los segmentos de vídeo codificados con el primero es menor. Los tiempos promedio de codificación son mayores que los indicados en la Tabla 5.3 (la fila con 8 réplicas) porque en este experimento codificamos cada segmento de vídeo en cuatro resoluciones. Además, los tiempos de subida también son mayores que los de la Tabla 5.3, porque ahora se suben cuatro vídeos codificados.



a)



b)

Figura 5.10: Consumo de CPU (en %) por número de CPU para cada nodo trabajador con un máximo de 8 réplicas fat de la función `vmaf-fn` cuando los segmentos se codifican usando: a) x264, b) AV1.

```
ffmpeg -i segmento1.mp4 \
        -filter_complex '[0:v]yadif,split=4[out1][out2][out3][out4]' \
        -map '[out1]' -vcodec libx264 -s 854x480 -b:v 1350k -maxrate 1500k -bufsize 3M \
           -threads 6 segmento1_480p.mp4 \
        -map '[out2]' -vcodec libx264 -s 1280x720 -b:v 2750k -maxrate 4M -bufsize 8M \
           -threads 6 segmento1_720p.mp4 \
        -map '[out3]' -vcodec libx264 -s 1920x1080 -b:v 6M -maxrate 8M -bufsize 16M \
           -threads 6 segmento1_1080p.mp4 \
        -map '[out4]' -vcodec libx264 -s 3840x2160 -b:v 11M -maxrate 14M -bufsize 28M \
           -threads 6 segmento1_2160p.mp4
```

Listing 7: Llamada de ejemplo realizada a ffmpeg en la función x264-fn para codificar un segmento a diferentes resoluciones y tasas de bits.

```
ffmpeg -i segmento1.mp4 \
        -filter_complex '[0:v]yadif,split=4[out1][out2][out3][out4]' \
        -map '[out1]' -vcodec librav1e -rav1e-params speed=10 -cpu-used 4 -row-mt 1 \
          -tile-columns 4 -threads 6 -s 854x480 -b:v 785k -maxrate 1.5M -bufsize 3.1M \
          segmento1_480p.mp4 \
        -map '[out2]' -vcodec librav1e -rav1e-params speed=10 -cpu-used 4 -row-mt 1 \
          -tile-columns 4 -threads 6 -s 1280x720 -b:v 1.4M -maxrate 2.8M -bufsize 5.6M \
          segmento1_720p.mp4 \
        -map '[out3]' -vcodec librav1e -rav1e-params speed=10 -cpu-used 4 -row-mt 1 \
          -tile-columns 4 -threads 6 -s 1920x1080 -b:v 2.7M -maxrate 5.5M -bufsize 11M \
          segmento1_1080p.mp4 \
        -map '[out4]' -vcodec librav1e -rav1e-params speed=10 -cpu-used 4 -row-mt 1 \
          -tile-columns 4 -threads 6 -s 3840x2160 -b:v 4.1M -maxrate 8.2M \
          -bufsize 16.5M segmento1_2160p.mp4
```

Listing 8: Llamada de ejemplo realizada a ffmpeg en la función av1-fn para codificar un segmento a diferentes resoluciones y tasas de bits.

Tabla 5.5: Tiempos promedio por segmento (en milisegundos) para descargar, codificar a cuatro resoluciones y subir todos los segmentos para diferentes vídeos, utilizando 8 réplicas fat. La última columna muestra el tiempo (en segundos) para completar la codificación de todos los segmentos de cada vídeo.

	Tiempos para x264			
Vídeo	***Descargar (ms)***	***Codificar (ms)***	***Subir (ms)***	***Total (s)***
ANC	416.8	14097.2	507.3	193.5
ELD	857.1	13792.5	520.6	192.3
IND	1105.7	14411.7	583.5	268.6
SKA	1143.6	13567.0	478.9	280.9
	Tiempos para AV1			
Vídeo	***Descargar (ms)***	***Codificar (ms)***	***Subir (ms)***	***Total (s)***
ANC	376.9	81363.8	200.7	1016.6
ELD	751.0	89861.0	282.8	1105.1
IND	663.7	93745.7	252.9	1555.7
SKA	795.6	84581.4	251.5	1528.7

5.3.5. Pipelines de procesamiento de contenido multimedia dirigidos por eventos

Para demostrar un *pipeline* simple de procesamiento de contenido dirigido por eventos, hemos añadido una bandera a los datos enviados a las funciones de codificación, `x264-fn` y `av1-fn`, para señalar si un *Cloud event* debe ser generado después del proceso de codificación para calcular el VMAF en la función `vmaf-fn`. En este caso, `x264-fn` y `av1-fn` actúan como *sinks* y fuentes de eventos.

La Figura 5.11 muestra un ejemplo de pipeline de codificación y obtención de la calidad:

- (1) la fuente de eventos solicita la codificación de un segmento utilizando x264,
- (2) este evento se procesa en la función x264 que descarga el vídeo y lo procesa.
- Cuando se sube el vídeo codificado, (3) se genera un evento para cal-

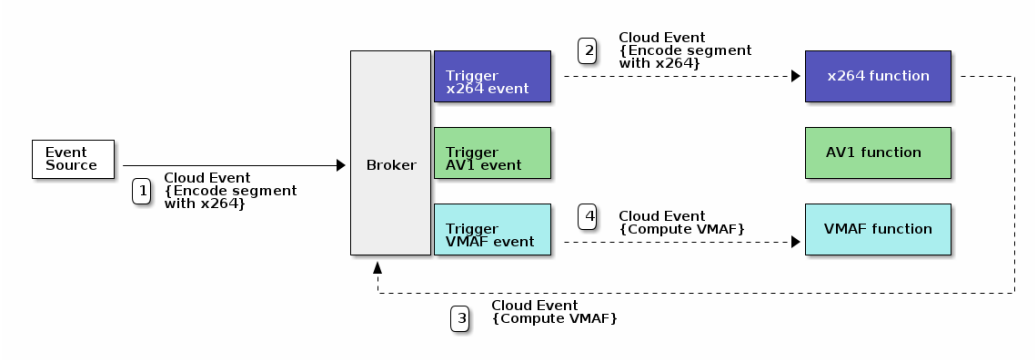


Figura 5.11: Ejemplo de pipeline para codificar un segmento de vídeo y obtener la calidad del VMAF.

cular el VMAF,

- Por último, si el flag `computeVMAF` es true, se lanza un evento que (4) inicia la función VMAF para calcular la métrica de calidad entre el vídeo original y el codificado.

Hemos codificado los 94 segmentos del vídeo ANC utilizando un máximo de 12 funciones que solicitan 2 CPUs cada una y utilizando un máximo de 6 funciones VMAF que solicitan 4 CPUs cada una (el número total de CPUs solicitadas es de 48). Al principio del experimento, no se ejecuta ninguna réplica de las funciones (arranque en frío).

Un esquema de una llamada de ejemplo que activa el *pipeline* se puede ver en el Listing 9.

En el cuerpo del *Cloud event*, solicitamos el cálculo de VMAF. Por lo tanto, después de que el vídeo haya sido codificado y subido a un servidor, se genera un *Cloud event* que se devuelve como respuesta. Este evento desencadena la llamada a la función VMAF. El cuerpo del *Cloud event* generado contiene información sobre el vídeo original (la referencia), el vídeo codificado (el distorsionado) y el servidor de subida para almacenar el cómputo del VMAF (ver Listing 10).

Cuando se ejecuta la función VMAF, la referencia y el vídeo distorsionado se descargan en paralelo, después se calcula el VMAF y el archivo JSON con los resultados se carga en el servidor de carga.

La Figura 5.12 muestra la programación de los trabajos en las réplicas

```
curl http://localhost:8080/http-event-source \
     -H 'Ce-Type: x264' \
     -H 'Ce-Source: /curl' \
     -H 'Ce-specversion: 1.0' \
     -H 'Ce-Id: 1' \
     -d '{"sourceUrl":"http://download-server/video/segmento1.mp4",
          "crf":24,
          "frameRate":"23.98",
          "extraParams":"-threads 2",
          "destUrl":"http://upload-server/upload",
          "videoNameDestination":"segmento1-crf-24.mp4",
          "destUrl":"http://upload-service/upload",
          "bucketParamName":"bucket",
          "bucketParamValue":"encodedVideos/"
          "computeVMAF":true,
          "extraVMAFParams":"model:n_threads=4"}'
```

Listing 9: Llamada de ejemplo que activa la *pipeline*.

```
Ce-Type: VMAF
Ce-Source: /x264-fn
Ce-specversion: 1.0
Ce-Id: 2
'{"referenceVideoURL":"http://download-server/video/segmento1.mp4",
 "distortedVideoURL":"http://upload-server/encodedVideos/segmento1-crf-
↪  24.mp4",
 "destUrl":"http://upload-server/upload",
 ...,
 "extraVMAFParams":"model:n_threads=4"}'
```

Listing 10: Ejemplo del cuerpo del evento para solicitar el cálculo de VMAF.

disponibles. El número máximo de réplicas es 12 para x264 y 6 para VMAF. La longitud del segmento es proporcional a la duración del trabajo y el número que aparece debajo de cada segmento es el número de segmento de vídeo que se procesa en la réplica correspondiente (indicado a la izquierda). Inicialmente, se generan 94 eventos para codificar todos los segmentos del vídeo ANC. Como se puede observar, se inician 12 réplicas para codificar los segmentos utilizando x264. Cuando una réplica termina la codificación, lanza un evento Cloud para calcular el VMAF del segmento recién codificado y recibe otro evento para codificar otro segmento.

Podemos observar que cuando los primeros trabajos de codificación terminan, las réplicas deben iniciarse para calcular el VMAF (hasta el máximo permitido de 6). En la Figura 5.12 se han resaltado dos trabajos. Cuando una réplica finaliza la codificación del segmento 7, dispara un evento Cloud para computar el VMAF y debe iniciarse una réplica con `vmaf-fn` (cold-start). Por el contrario, cuando otra réplica termina la codificación del segmento 62 y dispara el CloudEvent para calcular el VMAF, este evento es recibido inmediatamente por una réplica con `vmaf-fn` en ejecución (warm-start).

5.4. Comparación de propuestas de codificación de vídeo con serverless

En esta sección, vamos a hacer una comparación entre los trabajos anteriores y nuestra propuesta. Como hemos visto, en la Sección 2.5.3, los trabajos existentes, relacionados con el paradigma de la codificación de vídeo *serverless*, son muy diferentes de la arquitectura propuesta. Por esta razón, proporcionamos una comparación cualitativa para ver qué beneficios y limitaciones proporciona nuestra arquitectura en comparación con los trabajos anteriores, tal y como se resume en la Tabla 5.6. La información mostrada en la Tabla 5.6 ha sido obtenida de los trabajos estudiados en la sección de trabajos relacionados 2.5.3. Como se puede observar, hay algunas celdas con el código N/A, esto es debido a que no se ha podido encontrar la información en sus respectivos trabajos.

Nuestra propuesta solo puede compararse con (Fouladi et al., 2017) y (Ao et al., 2018b). Ambos trabajos anteriores utilizan la plataforma *serverless* de Amazon llamada AWS lambda para ejecutar sus propuestas. Por ello, disponen de una plataforma ilimitada de recursos pero con costes por ejecución

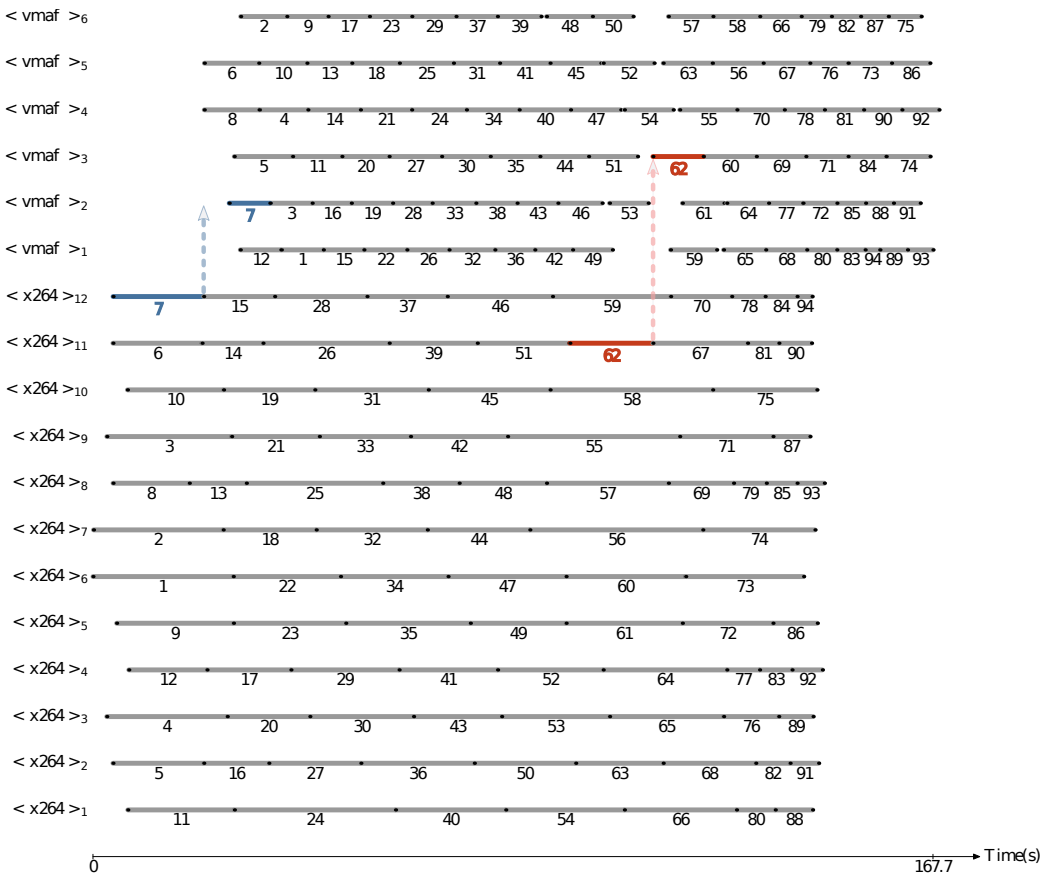


Figura 5.12: Secuenciación de trabajos en un proceso que comprende tanto la codificación como la evaluación de la calidad del vídeo codificado, utilizando para ello las funciones x264 y VMAF, respectivamente.

de trabajo. En nuestra propuesta, hemos utilizado Knative en nuestra plataforma, limitada en recursos, pero que nos permite tener el control total del sistema.

Mu framework es utilizado por (Fouladi et al., 2017) y (Ao et al., 2018b) para la gestión de ejecuciones paralelas, mientras que nuestra propuesta utiliza CloudEvents y la propia plataforma *serverless* de código abierto Knative. El uso del Mu framework es muy residual ya que el proyecto fue abandonado en 2018 y desde entonces no hay soporte.

Si analizamos en detalle los sistemas de codificación de vídeo presentados, existen grandes diferencias entre las propuestas anteriores y nuestro sistema. En concreto, el propuesto en (Fouladi et al., 2017) sólo soporta el códec VP8, que hoy en día no se utiliza debido principalmente a la compatibilidad que

ofrece la familia de códecs H264 (x264) y VP9 (evolución del VP8). Además, este códec no ofrece un buen rendimiento para resoluciones *ultra-high* como 4K, ya que VP9 o AV1 muestran un mejor rendimiento para esas resoluciones. En nuestra propuesta, hemos implementado 2 códecs: x264 y AV1. El primero, x264, se seleccionó porque es el códec más utilizado actualmente, aunque para 4K o resoluciones superiores hay códecs que funcionan mejor. AV1 se seleccionó porque es un códec de última generación. Está pensado para resoluciones *ultra-high*, como 4K, 8K y 16K. Nuestra propuesta y (Fouladi et al., 2017) se probó utilizando vídeos 4K, mientras que (Ao et al., 2018b) utiliza vídeos con resoluciones de 1080p y 720p.

Ligado a esto, si analizamos el tamaño del segmento utilizado, (Ao et al., 2018b) y nuestra propuesta utilizan tamaños de segmento de 2 segundos, mientras que (Fouladi et al., 2017) utiliza segmentos de 4 segundos divididos en micro-segmentos de seis fotogramas cada uno. En ambos casos, se utilizan tamaños de alrededor de 2 a 4 segundos, lo que supone un buen compromiso entre la eficiencia de la codificación y la flexibilidad para la adaptación del flujo a los cambios de ancho de banda (Lederer, 2020). Además, se detalla si se modifican los códecs de vídeo para la plataforma *serverless*. Por un lado, (Fouladi et al., 2017) necesita modificar el códec VP8 para que su propuesta funcione de forma adecuada, por otro lado, para (Ao et al., 2018b) y nuestra propuesta no es necesario debido a que las funciones de codificación de vídeo se basan en la herramienta FFmpeg.

Otro aspecto analizado en la Tabla 5.6 es la existencia de otras funciones distintas a la codificación. En este caso, (Ao et al., 2018b) introduce otras funciones en su sistema como la detección de caras. En nuestra propuesta, también se ha incluido una función de calidad de vídeo, donde se calcula el VMAF por segmento a partir de un vídeo de referencia y un vídeo codificado.

Según los trabajos, (Fouladi et al., 2017) no soporta *pipelines*, mientras que (Ao et al., 2018b) y nuestra propuesta muestran cómo construir *pipelines* y su rendimiento. El único trabajo que se propone para un entorno con recursos limitados es la propuesta presentada en este trabajo, mientras que las propuestas anteriores se desarrollaron en entornos donde la única limitación es el coste económico. Por último, nuestra propuesta muestra un estudio detallado sobre la escalabilidad y el rendimiento del sistema. Este tipo de estudio también fue realizado por (Ao et al., 2018b) pero el nivel de profundidad no fue tan alto, mientras que (Fouladi et al., 2017) presenta un estudio muy ligero sobre el rendimiento y escalabilidad de su propuesta.

Tabla 5.6: Comparación cualitativa de sistemas de codificación de vídeo basado en serverless

Características	**Fouladi, S. et al. (2017)**	**Ao, L. et al. (2018)**	**Our proposal (2022)**
Propiedad de la plataforma	Amazon	Amazon	Own Platform
Plataforma serverless	AWS lambda	AWS lambda	Knative solución Open-source
Framework para gestionar la ejecución paralela	Mu	Mu	CloudEvents
Códecs de vídeo	VP8	N/A	x264 and AV1
Resoluciones de vídeo probadas	4K	1080p and 720p	4K
Otras funciones serverless	No	Sí, incluye reconocimiento facial	Sí, calcula métrica de calidad VMAF
Tamaño del segmento de vídeo	4 segundos, divididos en 16 microsegmentos de 6 fotogramas cada uno	2 segundos	2 segundos
Códec modificado	Si	No, se basa en la herramienta ffmpeg	No, se basa en la herramienta ffmpeg
Propuestas de pipelines	No	Si	Si
Recursos limitados plataforma serverless	No	No	Si
Escalabilidad y estudio de rendimiento	Bajo	Medio	Alta

5.5. Conclusiones

Como podemos comprobar en nuestra vida cotidiana, el consumo de recursos multimedia a través de Internet tiene un gran peso. Películas, series y programas de televisión se consumen a través de plataformas comerciales que utilizan HAS como protocolo de *streaming*. Por ello, deben preparar los contenidos utilizando codificaciones lo más eficientes posibles. Esto lleva a paralelizar los trabajos y utilizar la Nube para ahorrar costes. Si pensamos que estas plataformas de *streaming* son capaces de generar miles de horas de contenidos al mes, se necesitan soluciones dinámicas y elásticas para realizar las tareas de codificación. Por ello, un sistema de codificación basado en el paradigma FaaS sobre una plataforma *serverless* podría ser una solución óptima.

En este trabajo, se han desarrollado tres funciones sin servidor basadas en eventos (x264, AV1 y VMAF) y se han encapsulado en tres pequeños contenedores. Se ha analizado su comportamiento cuando se despliegan en una plataforma sin servidor local con limitaciones Los experimentos analizan la escalabilidad y el consumo de recursos para varios estados del contenedor (arranque en frío y arranque en caliente), variando el número máximo de réplicas y los recursos asignados a las mismas (réplicas de función *fat* y *slim*).

Los resultados de las diferentes pruebas realizadas muestran el buen rendimiento de las funciones *serverless* propuestas escalando réplicas y distribuyendo los trabajos uniformemente entre todas las réplicas. Los resultados muestran que utilizando réplicas *slim*, el tiempo total de codificación se reduce en un 18 %, importante en las plataformas de pago por uso. Por otra parte, las réplicas *fat* serían más adecuadas en escenarios de *streaming* de vídeo en directo ya que el tiempo de codificación por segmento disminuye. Se ha demostrado que es posible realizar una codificación multirresolución de cada segmento por llamada a la función y esto se ha evaluado codificando cuatro vídeos a cuatro resoluciones con x264 y AV1.

Por último, hemos demostrado que es posible construir *pipelines* para realizar varias secuencias de cómputo encadenando CloudEvents. En este capítulo, hemos presentado un ejemplo de *pipeline* de eventos para codificar segmentos de vídeo y obtener su calidad a partir de un único evento. Este *pipeline* ha sido probado, y los resultados muestran una adecuada distribución y encadenamiento de los trabajos entre las réplicas disponibles de cada tipo de función.

Capítulo 6

VQMTK: Un software de código abierto para evaluar la calidad del vídeo

Cualquier tecnología suficientemente avanzada es indistinguible de la magia.

Arthur C. Clarke

Resumen:

El contenido de vídeo en Internet no deja de crecer. Es por ello, que las plataformas de *streaming* deben garantizar un cierto nivel de calidad a la hora de preparar sus contenidos. Con este fin, se han desarrollado una aplicación *open source* que agrupa varias métricas para evaluar la calidad del vídeo de manera sencilla. Este trabajo integra 14 métricas en un contenedor Docker, para así disponer de una herramienta multiplataforma, la cual hemos denominado *Video Quality Metric ToolKit* (VQMTK). La herramienta desarrollada ofrece una interfaz web a través de Jupyter notebooks y un script Bash que combina todas las métricas en una única herramienta. La herramienta ha sido testada sobre vídeos 4K, y las pruebas muestran la capacidad de obtener todas las métricas incluidas con un rendimiento adecuado. Esta herramienta podrá ser utilizada, tanto en entornos científicos como educativos.

Este capítulo está sustentado por el trabajo (Moina-Rivera et al., 2023c).

6.1. Introducción

Los avances en la compresión de vídeo (Bross et al., 2021a) y la disponibilidad de sistemas de transmisión eficientes y asequibles (Jiang et al., 2021) han llevado a que el tráfico de vídeo represente la mayor parte del tráfico total de Internet en los últimos años.

En este contexto, la evaluación de la calidad del vídeo es esencial para garantizar la calidad del servicio del vídeo proporcionado y mejorar la calidad de la experiencia del usuario final (Karam et al., 2009; Wu, 2015). Los sistemas de evaluación permiten a los proveedores de servicios de vídeo conocer mejor la transmisión de sus contenidos e identificar posibles problemas de calidad que puedan afectar la experiencia del usuario. Si se dispone de una buena calidad del vídeo, los proveedores pueden asegurar una experiencia más satisfactoria para los usuarios y de esta forma aumentar su lealtad. Por lo tanto, resulta necesario evaluar la calidad de los contenidos de vídeo a lo largo de todo el proceso de *streaming*, incluyendo el procesamiento y la transmisión.

El proceso de evaluación de la calidad del vídeo (Video Quality Assessment -VQA) cuenta con una gran variedad de métricas, con distintos niveles de rendimiento y complejidad. Estas métricas pueden clasificarse en subjetivas y objetivas. La evaluación subjetiva suele considerarse la más fiable y precisa, ya que se realizan experimentos controlados donde los vídeos son evaluados por personas. No obstante, las pruebas subjetivas son costosas, requieren mucho tiempo y no pueden integrarse directamente en un sistema práctico como métrica de optimización, ya que siempre se necesita la interacción humana para la obtención de la propia métrica.

Estas evaluaciones suelen producir un valor de MOS (por ejemplo, 1-malo, 2-pobre, 3-regular, 4-bueno, 5-excelente), que representa la valoración promedio de la calidad general del contenido de vídeo. También se puede obtener un DMOS (diferencia de MOS), en el que los resultados, que representan la diferencia entre dos valoraciones, se normalizan mediante su media y desviación típica antes de calcular el promedio.

Las métricas de evaluación objetiva de la calidad, algunas de ellas diseñadas y/o entrenadas a partir de datos de evaluación subjetiva, permiten predecir la calidad visual automáticamente y resultan ideales para evaluar y optimizar el rendimiento de los sistemas de transmisión en todas las etapas, incluida la preparación de contenido multimedia.

Las métricas objetivas de evaluación de la calidad del vídeo se pueden clasificar en tres categorías (Chikkerur et al., 2011):

- Las métricas objetivas VQA de referencia completa (FR) requieren acceso completo a la información del vídeo de referencia.

- Las métricas objetivas VQA de referencia reducida (RR) sólo necesitan un conjunto específico de características de vídeo de referencia y del vídeo distorsionado para evaluar la calidad.

- Las métricas objetivas VQA sin referencia (NR) no tienen acceso a la información de la secuencia de referencia y solo calculan la calidad del vídeo basándose en el vídeo distorsionado.

En resumen, es importante evaluar la calidad del contenido de vídeo en todas las etapas del proceso de transmisión, y para ello se han desarrollado tanto métricas subjetivas como objetivas de VQA.

Con esta motivación, en este capítulo se presenta la herramienta VQMTK, que integra 15 métricas objetivas VQA en una imagen de contenedor Docker para ponerlas a disposición como software de código abierto; la Tabla 6.1 muestra una comparativa de nuestra propuesta y los trabajos descritos en la subsección 2.5.4.

El contenido de este capítulo se estructura de la siguiente manera: En la Sección 2, se detalla la encapsulación de diversas tecnologías y herramientas en una imagen de contenedor, su arquitectura y los distintos modos en los que pueden ser ejecutadas. La Sección 3, por otro lado, se dedica a exponer la evaluación de la herramienta VQA en términos de consumo de recursos, en específico, la CPU y la memoria RAM, analizadas por métrica. Finalmente, la Sección 4 reúne las conclusiones derivadas de los hallazgos anteriores.

Tabla 6.1: Comparación de las características de `vqmtk` y otras herramientas de evaluación de la calidad de vídeo (N/A, significa que esta característica no está disponible).

Características	**VQMTK**	**VQMT Free**	**VQ Probe Free (3 / 14 días)**	**Video Quality Analyzer (2 semanas)**
		Funcionalidad flexible		
Valores de métrica por fotograma	PSNR, SSIM, VMAF, MSSIM, VIF, CAMBI	PSNR, SSIM, MSSSIM, NIQE, VQM, 3SSIM	PSNR, SSIM, VMAF, MSSIM, CAMBI, CIEDE2000	PSNR, PSNR-HVS, PSNR-HVS-M, SSIM, VMAF, MSSIM, VIF
Valor promedio por secuencia	✓	✓	✓	✓
Valores de métrica por componente de color	PSNR, SSIM	PSNR, SSIM, MSSIM	PSNR	PSNR
		Soporte HDR		
HDR	x	✓	N/A	N/A
		Métricas de calidad de vídeo soportadas		
APSNR	✓	x	x	x
BRISQUE	✓	x	x	x
CAMBI	✓	x	✓	x
CIEDE2000	✓	x	✓	x
MS-SSIM	✓	✓	✓	✓
NIQE	✓	✓	x	x
PSNR	✓	✓	✓	✓
PSNRHVS	✓	x	x	✓
SSIM	✓	✓	✓	✓
STRRED	✓	x	x	x
VIF	✓	x	x	✓
VIIDEO	✓	x	x	x
VMAF	✓	✓	✓	✓
VQM	✓	✓	x	x
3SSIM	x	✓	x	x
		Códecs de vídeo		
AVC	✓	✓	✓	✓
HVEC	✓	✓	✓	✓
VP9	✓	✓	x	x
AV1	✓	✓	✓	✓
		Contenedores de vídeo compatibles		
.MP4	✓	✓	✓	✓
.MKV	✓	✓	✓	✓
.WEBM	✓	✓	✓	✓
RAW VIDEO (.Y4M)	✓	✓	✓	✓
		Resolución y soporte de profundidad de bits		
Resolución de vídeo compatible	Todas las resoluciones (480p, 720p, 1080p, 1440p, 2160p, puede ser personalizado)	Until 720p	Until 1080p	Todas las resoluciones (480p, 720p, 1080p, 1440p, 2160p, puede ser personalizado)
Video with 10, 12, 14, 16-bit RGB	✓	✓	✓	✓
		Modos interfaz		
GUI	✓	✓	✓	✓
CLI	✓	x	x	x
		Soporte de formato de datos		
json	✓	✓	x	x
csv	✓	✓	x	x
xml	✓	x	x	x
		Análisis comparativo		
Número de referencias que puede ser comparado con el vídeo original simultaneamente.	1 Canal	2 Canales	9 Canales	1 Canal

6.2. Integración de las métricas VQA en una imagen de contenedor

Con el fin de proporcionar cada una de las métricas de calidad de vídeo indicadas en el apartado anterior como si fuera una única herramienta, en esta sección se describen las tecnologías y herramientas que se han utilizado para integrarlas. Además, se analizan los distintos contextos en los que se puede ejecutar una métrica VQA.

6.2.1. Arquitectura de componentes

Con el objetivo de integrar las métricas de calidad, descritas en la sección anterior, se utilizan proyectos de terceros que las implementan. Además, de proyectos para procesar (descomprimir, segmentar, transcodificar, etc) los vídeos analizar. Estos proyectos proporcionan un artefacto/paquete como parte de su solución. En la Figura 6.1 se muestra un diagrama de componentes de los artefactos y componentes que integran esta solución.

El artefacto `FFmpeg`[10] permite procesar el contenido de vídeo (p. ej. codecs como H264, H265, VP9, AVI, entre otros, de vídeo para comprimir, descomprimir, etc.) y calcular un conjunto de métricas de calidad como SSIM, PSNR; VMAF al integrar libvmaf en la construcción. El artefacto `vmaf`[13] integra un conjunto de métricas de calidad como la propia VMAF (en un paquete python), MSSIM, VIF o CAMBI. El artefacto `scikit-video`[15] proporciona las métricas BRISQUE, NIQE, VIIDEO, STRRED. El artefacto `vqmetric`[16] implementa la métrica VQM y por último el artefacto `av-metrics`[11] integra el conjunto de métricas APSNR, PSNR-HVS y CIEDE200.Por otro lado, se integra el artefacto `dash`[1] de Plotly con el objetivo de proporcionar una herramienta para poder ilustrar, por ejemplo, el resultado de experimentos. En la imagen del contenedor disponible en `https://hub.docker.com/r/cloudmedialab/vqmtk` también se integra un artefacto para obtener el SITI.

La solución contempla dos componentes principales que hacen uso de estos artefactos; por una parte esta el componente `Jupyter`[2], que proporciona una interfaz gráfica de usuario (GUI) vía navegador Web (el usuario accede

[1] `https://dash.plotly.com`
[2] `https://jupyter.org`

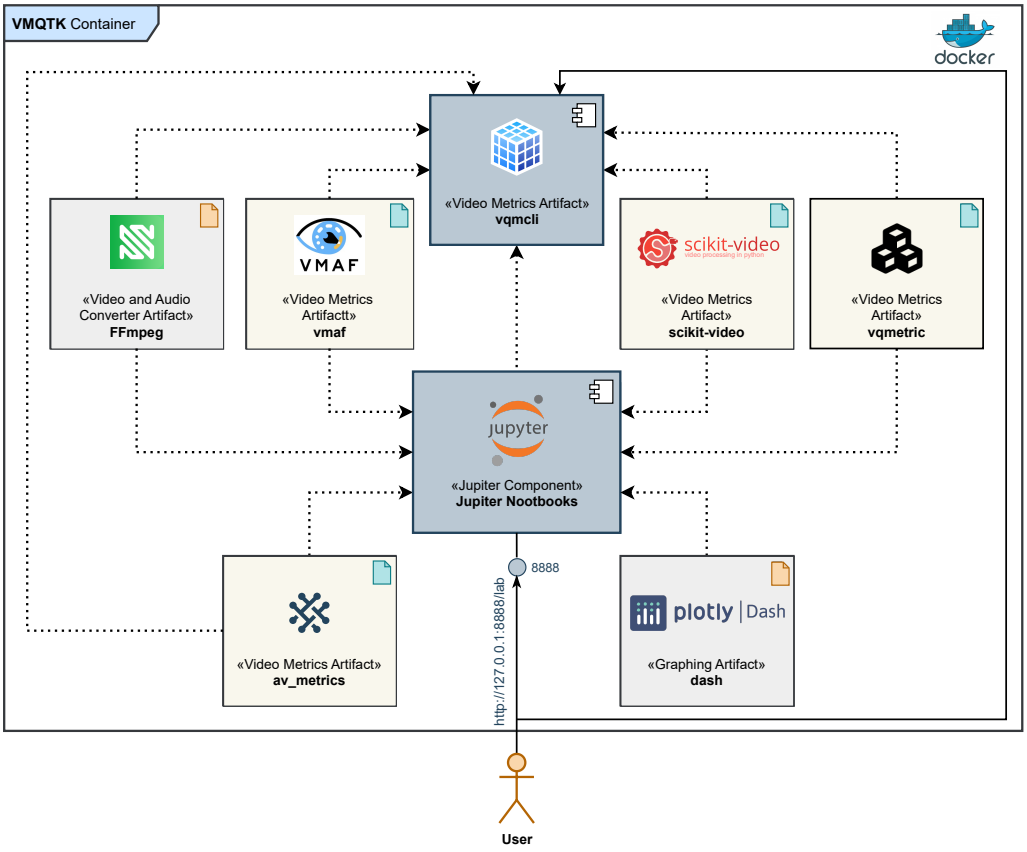


Figura 6.1: Arquitectura basada en componentes.

desde un navegador) para ejecutar cada una de las métricas indicadas previamente; por otra parte, el componente `vqmcli`, que utiliza todos los artefactos para mostrarlos como una sola herramienta (el usuario utiliza la interfaz de línea de comandos (CLI) para calcular una o varias métricas a la vez); a su vez, este componente puede ser ejecutado desde `Jupyter`.

La descripción, la implementación y el uso de estos componentes se muestra en las secciones siguientes. Esta solución se integra en un contenedor Docker, como se muestra a continuación.

6.2.2. Containerización

Para aprovechar las ventajas de la containerización, como es la capacidad de empaquetar código de software y sus dependencias como si se tratara

de una aplicación portátil y multiplataforma, se utilizó la plataforma Docker como tecnología para integrar todos los tipos de VQA en una imagen Docker —las imágenes se convierten en contenedores cuando las ejecuta un *container runtime*—, así como un entorno interactivo basado en la web con la integración de Jupyter Notebook.

Para construir automáticamente la imagen del contenedor, Docker permite definir un documento de texto llamado Dockerfile. Este documento contiene todas las instrucciones sucesivas que un usuario podría llamar en la línea de comandos para construir una imagen. Esto nos permitió definir una construcción/instalación automatizada de cada una de las métricas así como de la herramienta web, sin embargo, debido a la variedad de dependencias entre los diferentes proyectos utilizados, los archivos y herramientas descargadas sólo para el tiempo de compilación, resultó en una imagen de contenedor con un tamaño del orden de 6 GB. Por lo tanto, para mitigar este problema y obtener una imagen más pequeña, se utilizaron compilaciones Docker muti-etapa.

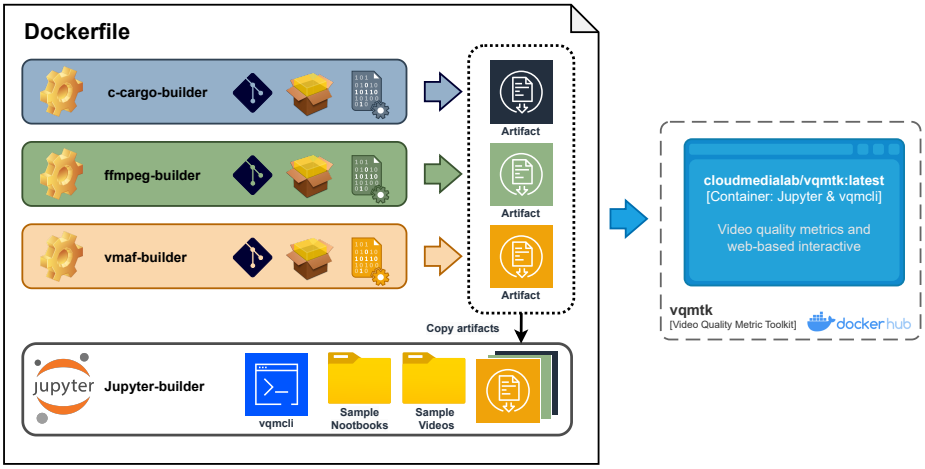


Figura 6.2: Dockerfile usando el multi-etapa para construir la imagen de contenedor `vqmtk`.

Con la compilación multi-etapa (cada etapa ejecuta una imagen de contenedor), esto permite copiar selectivamente los artefactos de una etapa a otra, dejando atrás todo lo que no se desea en la imagen final. La Figura 6.2 muestra una ilustración de alto nivel de las etapas establecidas para la imagen final.

Por otra parte, el Dockerfile que se proporciona en el repositorio de este

proyecto [3], para reproducir la construcción de la imagen final o hacer cambios o mejoras según sea necesario. La imagen del contenedor está disponible en docker hub [4].

6.2.3. Modos para ejecutar una evaluación de calidad de vídeo

Con las métricas VQA integradas en una imagen de contenedor Docker, pueden ejecutarse desplegando la imagen como un contenedor con `Docker Engine`. A continuación se indican las formas de ejecutar una métrica de calidad de vídeo:

6.2.3.1. Modo: Command Line Interface (CLI)

Un VQA puede ser ejecutada de dos maneras: desde una línea de comandos (CLI) usando el comando `docker run` con la sentencia de comando correspondiente como argumento, o a través de un proceso de terminal (bash o shell) dentro del contenedor. Para ejecutar un VQA desde la CLI, se puede utilizar el comando de ejemplo mostrado en Listing 11-a. Los resultados se mostrarán en la consola del terminal como salida estándar (STDOUT), pero también pueden ser redirigidos a un archivo para su almacenamiento. Si se opta por ejecutar una VQA desde la terminal dentro del contenedor, para almacenar los resultados se deberá utilizar uno de los tipos de montaje[5] proporcionados por Docker entre la máquina anfitriona y el contenedor, ver el ejemplo 2 del Listing 11-b.

6.2.3.2. Modo: Interfaz gráfica de Usuario (GUI)

Este proyecto pretende proporcionar el mayor número posible de métricas de calidad de vídeo mediante la integración de soluciones de software de terceros, y para calcular una métrica concreta es necesario seguir las instrucciones y limitaciones de dicha implementación y de la propia métrica.

Bajo esta premisa, para mostrar una llamada sencilla de cada una de las

[3] https://github.com/cloudmedialab-uv/vqmtk
[4] https://hub.docker.com/r/cloudmedialab/vqmtk
[5] https://docs.docker.com/storage/volumes/

```
# a) Llamada de ejemplo para calcular PSNR pasando el comando como argumento a docker
↪  run (llamada realizada en la maquina anfitriona).

docker run --rm -v /pathVideos/:/videos vqmtk:latest ffmpeg \
-i /videos/video-ref.mp4 -i /videos/video-dis.mp4 -lavfi "[0:v][1:v]psnr" -f null -

# b) Llamada de ejemplo añadiendo un punto de montaje para almacenar los resultados (la
↪  llamada proporciona una terminal bash dentro del contenedor).
docker run --rm -it -u $(id -u):$(id -u) -v /pathVideos/:/videos \
-v /pathResults/:/results vqmtk:latest bash
```

Listing 11: Llamada de ejemplo para ejecutar una métrica de calidad de vídeo desde una terminal del anfitrión o el contenedor.

métricas y proporcionar un entorno de trabajo accesible desde una interfaz web, se integra el proyecto Jupyter como parte de la imagen contenedora (última etapa en la construcción de la imagen Docker).

Este proyecto ofrece Jupyter Notebook (antes IPython Notebook) como aplicación web para crear y compartir notebooks, ofreciendo una experiencia de usuario simple y adecuada, que permite a los usuarios configurar y organizar flujos de trabajo de manera sencilla[2]

Jupyter puede equiparse con kernels para varios lenguajes diferentes (Julia, R, C++, Scheme, Ruby), sin embargo para este caso, sólo se requieren las versiones 2.7 y 3.9 de python (que hay que seleccionar dependiendo de los requisitos de la métrica a calcular), véase la sección B de la Figura 6.3.

Para proporcionar las llamadas de ejemplo, se crea un documento Jupyter Notebook[6] para cada una de las métricas VQA. Estos Notebooks están almacenados como parte del contenedor en el directorio `/home/jovyan/work/examples/`, véase la sección A de la Figura 6.3.

Además, se integra un segmento de 2 segundos de duración (50 fotogramas) del vídeo BigBugBunny[7], transcodificado a diferentes resoluciones y valores de Constant Rate Factor (CRF)[8] para ejecutar cada uno de los ejemplos.

[6] El documento Jupyter Notebook es un REPL (Read-Eval-Print Loop) ejecutado desde un navegador que contiene una lista ordenada de celdas de entrada/salida que pueden contener código, texto (usando Markdown), matemáticas, gráficos y rich media. Por debajo de la interfaz, un cuaderno es un documento JSON, siguiendo un esquema versionado, por lo general termina con la extensión ".ipynb".

[7] https://peach.blender.org

[8] https://slhck.info/video/2017/02/24/crf-guide.html

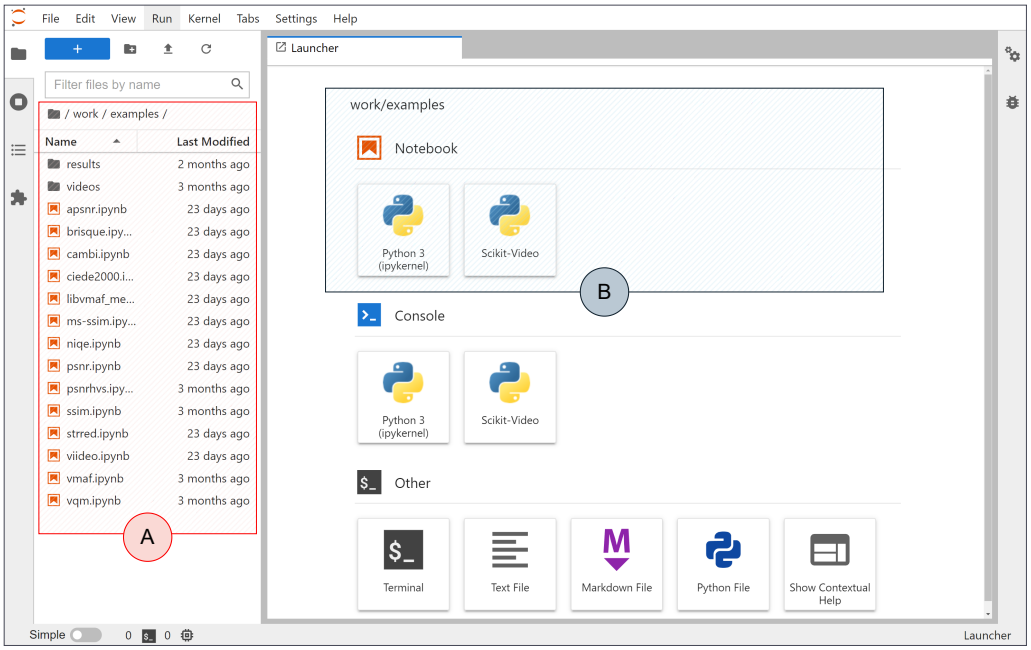


Figura 6.3: Sección A: Directorio con Notebooks de ejemplo. Sección B: Kernels disponibles.

Además del comando/script de ejemplo sobre cómo ejecutar una métrica VQA, véase la sección A en la Figura 6.5, para varios de los ejemplos se integra un script para hacer una gráfica de los valores de calidad por fotograma, utilizando el paquete gráfico `dash`, véase la sección B en la Figura 6.5.

El Listing 12-a muestra las instrucciones para desplegar Jupyter localmente y ejecutar los casos de ejemplo.

6.2.3.3. Modo: Herramienta vqmcli

Con el uso de los Notebooksse ha proporcionado un ejemplo de una llamada sencilla por métrica VQA. Esto permite al usuario probar y explorar las posibilidades de cada implementación para adaptarla a su caso de uso. No obstante, ajustarse a las limitaciones (soporte) de cada métrica, como el tamaño mínimo o máximo del segmento, la resolución, el tipo de códec, el tipo de contenedor, el submuestreo de croma, etc., puede dar lugar a tareas de preprocesado de vídeo (transcodificación, segmentación, etc.) sin que se altere la calidad de los videos utilizados para la evaluación.

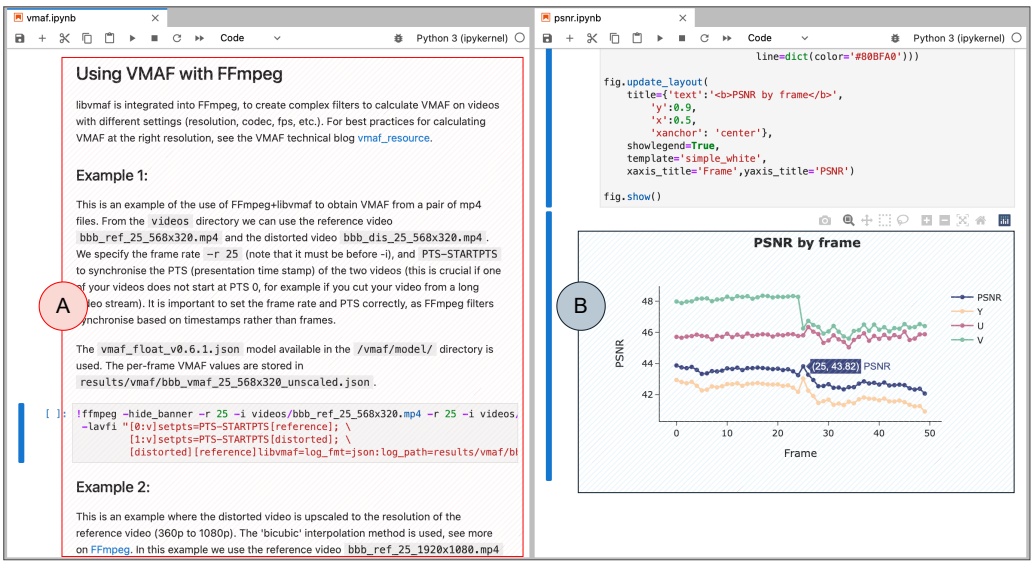


Figura 6.4: Sección A: Notebook de VMAF . Sección B: Representación gráfica de los valores PSNR por fotograma.

Además, el encadenar (secuencialmente) trabajos de VQAs pertenecientes a una misma implementación conlleva a repetir tareas que comparten, por ejemplo, descomprimir y cargar el vídeo en memoria, generando un tiempo de procesamiento innecesario. Por esta razón, la herramienta `vqmcli`, escrita en bash script, fue desarrollada y permite:

- Realizar un trabajo de preprocesamiento del vídeo de origen y de referencia, si la(s) métrica(s) lo requiere(n).
- Calcular una o más métricas utilizando una única instrucción de comando.
- Obtener los valores de calidad por fotograma de las implementaciones que lo soporten.
- Obtener un fichero con los resultados de calidad por VQA y segmento de vídeo.
- Especificar el tamaño del segmento dentro de un rango permitido, esto es necesario para evitar volcados de memoria.
- Soporte para múltiples contenedores y codecs (ver Tabla 6.2).

```
# a) Llamada de ejemplo para ejecutar el contenedor con Jupyter utilizando JupyterLabs y
↪  evaluar una VQA.
docker run --rm -it  -p 8888:8888 -p 8050:8050 -e JUPYTER_ENABLE_LAB=yes vqmtk:latest

# b) Imprimir la ayuda de la herramienta "vqmcli" usando "docker run".
docker run --rm vqmtk:latest vqmcli --help

# b) Llamada de ejemplo para calcular las métricas SSIM y PSNR utilizando la herramienta
↪  "vqmcli" desde la terminal del contenedor.

vqmcli -r video-ref.mp4 -d video-dis.mp4 -o /results/my_results --size 2 --psnr --ssim
↪  --ext json
```

Listing 12: Ejemplos de llamada para ejecutar un VQA tanto desde la interfaz web como desde la herramienta `vqmcli`.

Tabla 6.2: Códecs de vídeo admitidos por contenedor para cada métrica de calidad.

Contenedor	**MP4**				**MKV**				**WEBM**		**Y4M**
Códec	***H.264***	***H.265***	***VP9***	***AV1***	***H.264***	***H.265***	***VP9***	***AV1***	***VP9***	***AV1***	***RAW***
APSNR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BRISQUE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CAMBI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CIEDE2000	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSSSIM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NIQE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PSNR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PSNRHVS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SSIM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
STRRED	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VIF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VIIDEO	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VMAF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VQM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

El código de la herramienta ha sido liberado bajo los términos y condiciones descritos en el repositorio[3].

6.3. Experimentos y Evaluación del Rendimiento

En esta sección describimos las pruebas realizadas para validar la implementación de cada uno de las VQA. Hemos realizado experimentos para validar el soporte de codecs y contenedores de vídeo y el comportamiento

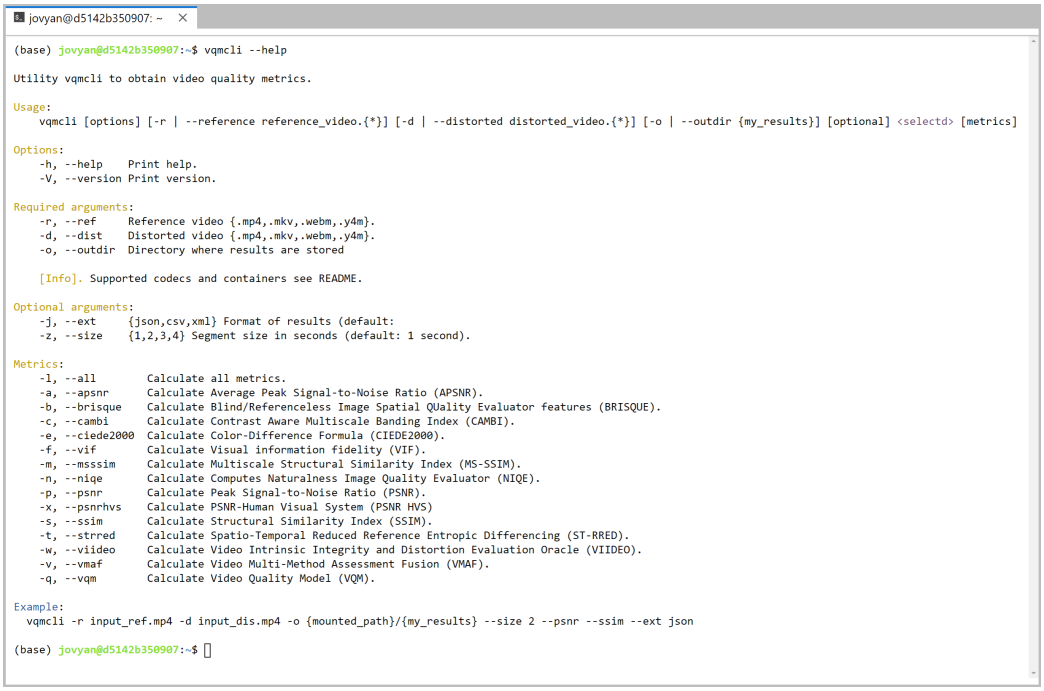


Figura 6.5: Ayuda de la herramienta `vqmcli` incluida en la imagen del contenedor.

de la herramienta `vqmcli` desarrollada en términos de consumo de recursos (CPU y memoria).

6.3.1. Información del Entorno de Pruebas

En nuestra evaluación del rendimiento, se han realizado varias pruebas con cuatro vídeos de libre acceso: Beauty (BEA), Bosphorus (BOS), Cityalley (CIT) y Honeybee (HON) del conjunto de datos UVG (Mercat et al., 2020). Estos vídeos se descargaron en resolución 2160p o 4K, con una profundidad de color de 8 bits, en formato YUV sobre el contenedor Y4M, y se almacenaron en un contenedor MP4 después de realizar una codificación con x264 en alta calidad (CRF=0, que genera vídeo codificado visualmente sin pérdidas). Los tamaños mostrados en la Tabla 6.3 son el tamaño final tras la codificación, y su complejidad espacial y temporal se muestra en la Figura 6.6.

En este trabajo, cada uno de estos vídeos ha sido codificado utilizando

Tabla 6.3: Secuencias de vídeo utilizadas para la evaluación y validación de la herramienta.

Vídeo	Resolución	FPS	Frames	Tamaño (GB)	Movimiento (Tipo / Cantidad)
BEA	3840x2160	30	600	4.1	Lento / Mucho
BOS	3840x2160	30	600	2.8	Lento / Mucho
HON	3840x2160	30	600	3.5	Lento / Poco
CIT	3840x2160	50	600	3.0	Rápido / Poco

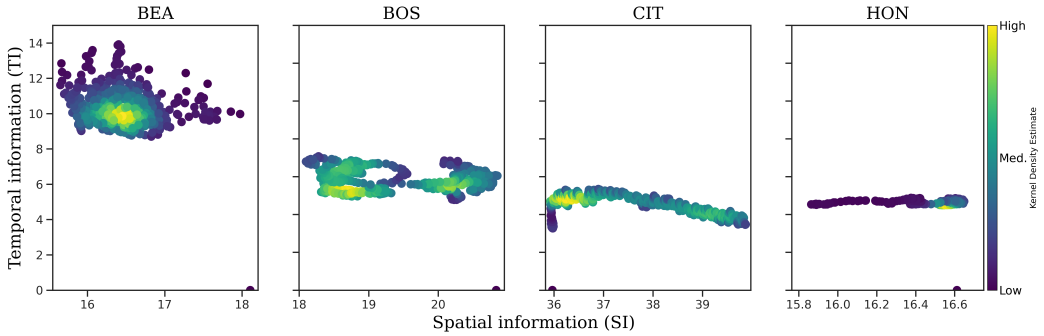


Figura 6.6: Mapa de calor de la información espacio-temporal (SI-TI) de todos los fotogramas.

x264[9], VP9[10] y AV1[11], y por codec dos valores de CRF. Obteniendo así 24 representaciones con diferentes valores de calidad. Para x264 los CRF elegidos fueron 23 y 40, VP9 los CRF fueron 30 y 50, y para AV1 los mismos que para VP9.

En nuestros experimentos, hemos seleccionado un tamaño de segmento de 2 segundos como entrada de `vqmcli` en todos los experimentos, para seguir el buen compromiso entre la eficiencia de codificación y la flexibilidad para la adaptación del flujo a los cambios de ancho de banda (Lederer, 2015).

Los experimentos se realizaron en una máquina virtual con 58 vCPUs, 24 GB RAM, Ubuntu 20.04 x64 bits y Docker Versión 19.03.5. La máquina anfitriona cuenta con 2 CPUs Intel Xeon Silver 4216 @2.10GHz con 16 núcleos cada una, 128 GB RAM @2666 MHz y disco WD40NDZW-11A8JS1 5400 RPM.

En todos los experimentos, no se ejecutó ningún otro proceso en la má-

[9] ffmpeg con codec library libx264
[10] ffmpeg con codec library libvpx-vp9
[11] ffmpeg con codec library libaom-av1

```
# Llamada para calcular una VQA utilizando la herramienta vqmcli, en este ejemplo VMAF.

docker run -u $(id -u):$(id -u) -v /pathVideos/:/videos -v /pathResults/:/results \
vqmcli:latest vqmcli -r /videos/video-ref.mp4 -d /videos/video-dis-crf-codec.mp4 \
-o /results/vmaf --size 2 --vmaf --ext json
```

Listing 13: Ejemplo para calcular una VQA con la herramienta `vqmcli`.

quina virtual para que las medidas de rendimiento fueran lo más precisas posibles.

6.3.2. Consumo de recursos

En esta sección, analizamos el uso de CPU y el consumo de RAM al calcular todas las métricas VQA utilizando la herramienta `vqmcli`. Para estos experimentos, se tuvieron en cuenta las siguientes consideraciones:

- Ejecutar un proceso `docker run` para calcular las diferentes combinaciones de VQA, vídeo, CRF y codec (2 códecs $\times$ 4 $\times$ vídeos $\times$ 2 CRFs $\times$ 14 métricas, en total 224 invocaciones secuenciales).
- Ejecutar un único proceso `docker run` por combinación.

En todos los experimentos, se utiliza la instrucción que se muestra en el Listing 13.

6.3.2.1. vqmcli usando x264

La Figura 6.7 despliega un diagrama de caja que detalla el consumo de CPU en porcentaje por métrica, basándose en datos de vídeos codificados con el códec x264 en sus distintas representaciones. En este conjunto, MSSIM se distingue como la métrica de mayor uso de CPU, con un promedio del 40.02 %. En contraste, las métricas BRISQUE, CAMBI, NIQE, VIIDEO y VQM requieren de un menor uso de CPU, registrando un porcentaje entre el 9 % y el 10 %.

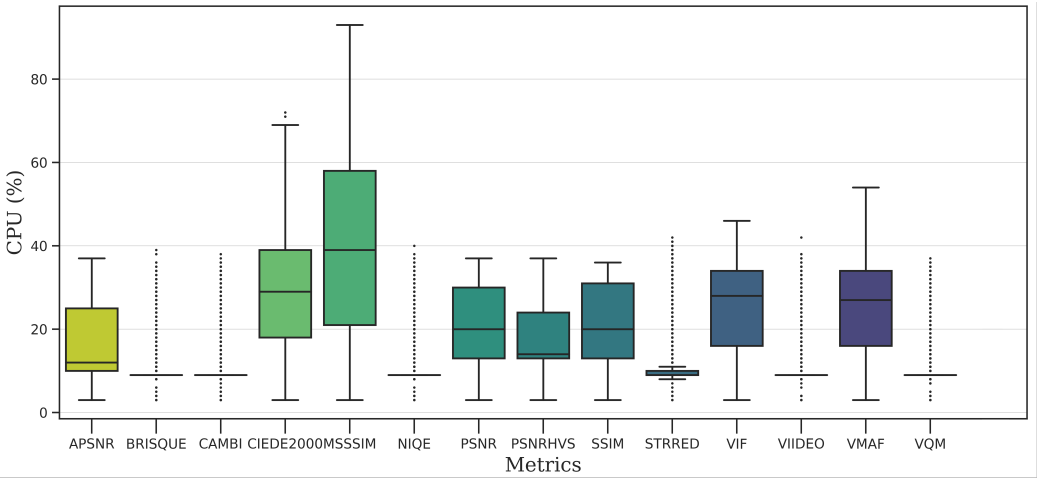


Figura 6.7: Consumo de CPU (en %) por métrica al calcular la calidad de los vídeos codificados con x264.

La Figura 6.8 ilustra el consumo de memoria atribuible a cada métrica a lo largo de los experimentos realizados. MSSIM se destaca como la métrica de mayor demanda de memoria, promediando un consumo de 7.51 GB. En contraposición, la métrica PSNRHVS muestra un consumo más moderado, con una media de 1.82 GB.

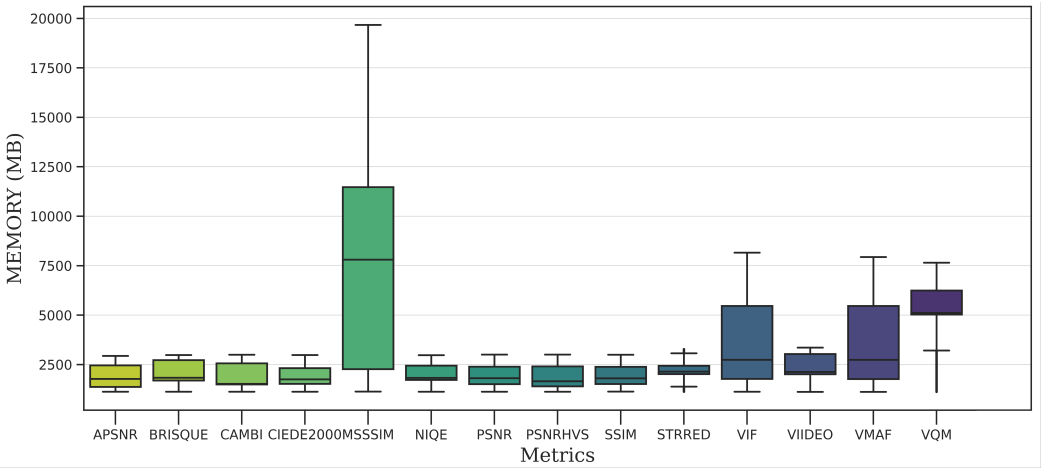


Figura 6.8: Consumo de memoria (en MB) por métrica al calcular la calidad de los vídeos codificados con x264.

Es importante resaltar que, entre las métricas evaluadas, VIIDEO demanda el mayor tiempo de cálculo en promedio, alcanzando los 154.72 minutos.

Por otro lado, SSIM resulta ser la métrica más eficiente en términos de tiempo, requiriendo apenas 1.78 minutos en promedio.

6.3.2.2. vqmcli usando VP9

La Figura 6.9 presenta un diagrama de caja (boxplot) que ilustra el porcentaje de consumo de CPU (%) por métrica, utilizando datos de vídeos codificados con el códec VP9 en diversas representaciones. Una vez más, se destaca MSSIM como la métrica de mayor demanda de CPU, exhibiendo una media de 39.46 %. Por otro lado, las métricas que reflejan un consumo más moderado de CPU, oscilando entre el 9 % y el 10 %, incluyen a BRISQUE CAMBI, NIQE, VIIDEO, y VQA.

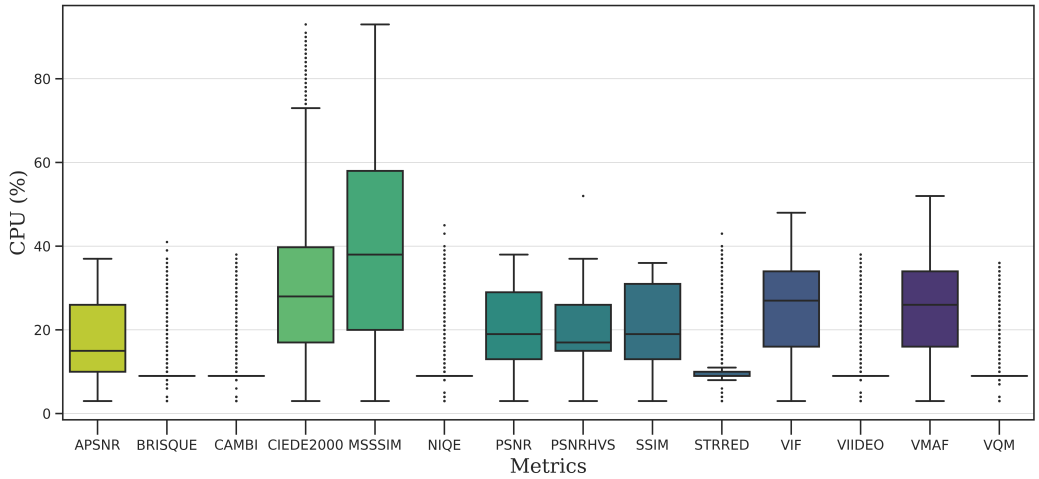


Figura 6.9: Consumo de CPU (en %) por métrica al calcular la calidad de los vídeos codificados con VP9.

La Figura 6.10 exhibe el consumo de memoria asociado a cada métrica a lo largo de los experimentos llevados a cabo. Se observa que MSSIM es la métrica de mayor demanda de memoria, con un promedio de 7.41 GB. En contraposición, APSNR presenta un consumo más contenido, registrando una media de 1.7 GB.

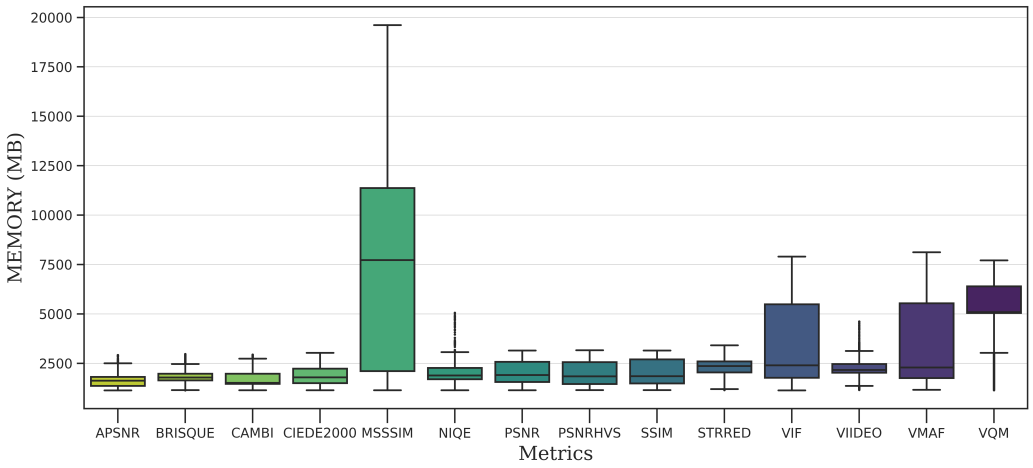


Figura 6.10: Consumo de memoria (en MB) por métrica al calcular la calidad de los vídeos codificados con VP9.

Además, de entre las métricas evaluadas, VIIDEO se distingue por demandar el mayor tiempo de cálculo, alcanzando los 160.16 minutos en promedio. En contraposición, SSIM sobresale por su eficiencia temporal, requiriendo apenas 1.84 minutos en promedio.

6.3.2.3. vqmcli usando AV1

La Figura 6.11 presenta un diagrama de caja que detalla el consumo de CPU en porcentaje por cada métrica, utilizando como base datos de vídeos codificados con el códec AV1 en sus diferentes representaciones. MSSIM se distingue por ser la métrica de mayor uso de CPU, promediando un consumo del 40.02 %. En contraste, las métricas BRISQUE, VIIDEO, CAMBI, NIQE y VQM muestran un menor requerimiento de CPU, registrando porcentajes de consumo entre el 9 % y el 10 %.

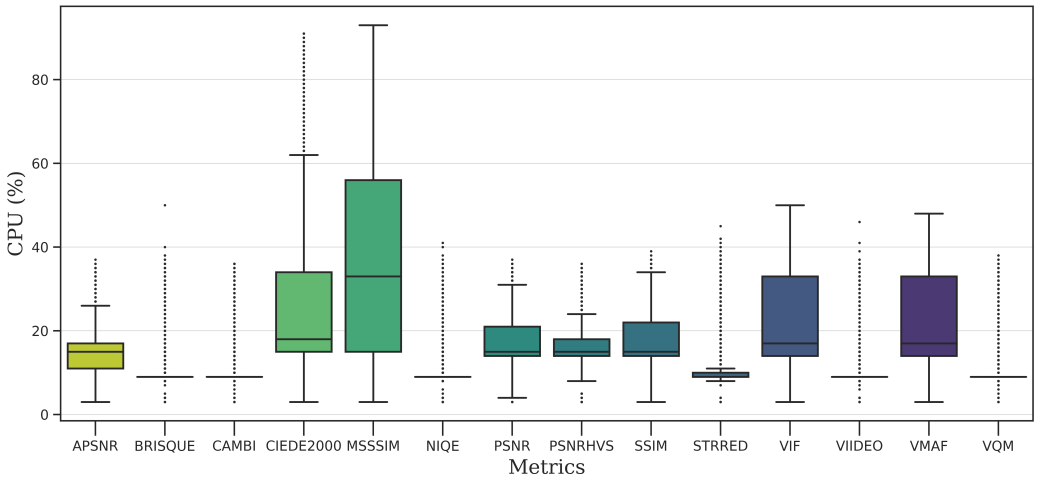


Figura 6.11: Consumo de CPU (en %) por métrica al calcular la calidad de los vídeos codificados con AV1.

La Figura 6.12 ilustra la demanda de memoria correspondiente a cada métrica a lo largo de los experimentos realizados. Se observa que MSSIM es la métrica que más memoria consume, con un promedio de 6.32 GB. Por otro lado, CAMBI destaca por su menor demanda de memoria, registrando una media de tan solo 1.54 GB.

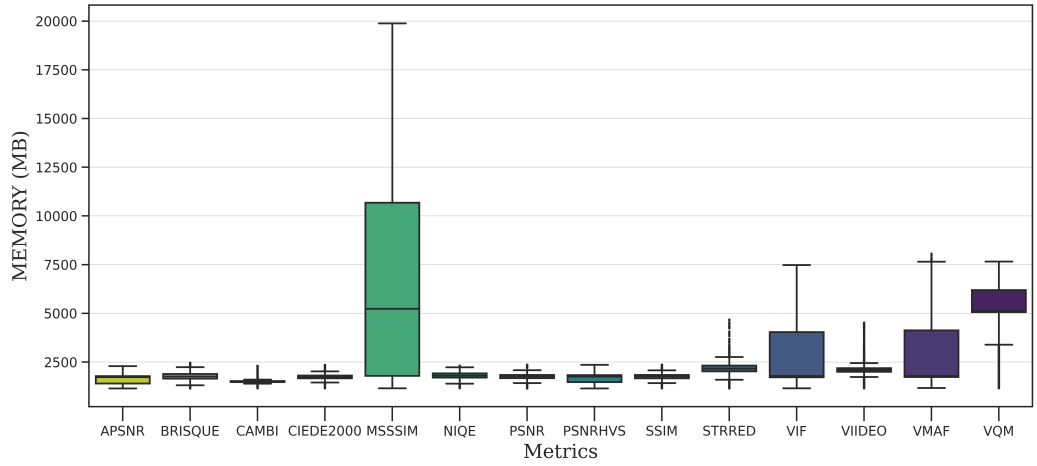


Figura 6.12: Consumo de memoria (en MB) por métrica al calcular la calidad de los vídeos codificados con AV1.

De igual forma, entre las métricas evaluadas, VIIDEO destaca por demandar el mayor tiempo de cálculo, con un promedio de 158.78 minutos.

Contrariamente, SSIM sobresale por su eficiencia en términos de tiempo, requiriendo solamente 2.98 minutos en promedio.

En la Tabla 6.4, se ofrece un resumen del tiempo promedio que la herramienta empleó para calcular la calidad en función de la métrica y el códec correspondientes a cada segmento de 2 segundos de todos los videos analizados.

Tabla 6.4: Tiempo (en minutos) utilizado para el análisis de calidad por métrica y por códec de vídeo.

Codec	**APSNR**	**BRISQUE**	**CAMBI**	**CIEDE2000**	**MSSSIM**	**NIQE**	**PSNR**	**PSNRHVS**
x264	2.85	19.80	134.97	3.03	5.05	30.61	1.81	2.87
VP9	2.44	19.52	141.57	2.79	5.15	30.52	1.87	2.48
AV1	3.60	20.48	142.16	3.91	6.24	31.56	3.03	3.63
Codec	**SSIM**	**STRRED**	**VIF**	**VIIDEO**	**VMAF**	**VQM**	-	-
x264	1.78	68.59	3.05	154.72	3.12	70.49	-	-
VP9	1.84	68.69	3.13	160.16	3.20	70.61	-	-
AV1	2.98	69.72	4.29	158.78	4.35	71.56	-	-

6.4. Conclusiones

La cantidad de contenido multimedia en Internet, especialmente de vídeo, continúa creciendo a un ritmo sin precedentes. Como resultado, las plataformas de transmisión deben garantizar un cierto nivel de calidad al preparar su contenido, particularmente el vídeo bajo demanda. Con este fin, a lo largo de los años la comunidad científica ha desarrollado varias métricas para evaluar la calidad del vídeo, tanto para uso gratuito como comercial. Dada la diversidad de métricas disponibles, este trabajo integra muchas de ellas en una imagen de contenedor Docker para crear una herramienta multiplataforma, VQMTK. La herramienta consta de una interfaz web fácil de usar y un script Bash que combina 14 métricas de calidad de vídeo y valores SI-TI en una única herramienta. Con VQMTK, desarrolladores e investigadores pueden evaluar fácilmente la calidad de distintas configuraciones de codificación de vídeo desde la fase inicial del proceso de transmisión. Esto puede orientar la selección de codificadores, sus ajustes y el conjunto de representaciones ofrecidas para mejorar la experiencia del usuario.

Aunque este trabajo no se centra en la optimización del rendimiento de cada métrica, se presentan pruebas de rendimiento por métrica VQA. Estos resultados pueden ser útiles para los usuarios de la herramienta a la hora de

asignar recursos para el entorno de ejecución (por ejemplo, si la herramienta se va a utilizar en una instancia en la Nube).

Además, podría usarse para generar conjuntos de datos para entrenar Redes Neuronales Profundas (Deep Neural Networks – DNN) y desarrollar otras soluciones de codificación de vídeo basadas en DNN (Micó-Enguídanos et al., 2023).

Finalmente, esta herramienta también podría utilizarse en entornos educativos, donde se traten temas de codificación y calidad de vídeos, ya que podría ser muy útil para el alumnado por su facilidad de despliegue y utilización.

Capítulo 7

Conclusiones y Trabajo Futuro

El aprendizaje es un tesoro que seguirá a su dueño a todas partes

Confucio (孔夫子)

Resumen:

En este capítulo, se presentarán las principales conclusiones de esta tesis y se discutirán los trabajos futuros relacionados con ellas. A lo largo de este trabajo, se han presentado resúmenes y conclusiones sobre los diferentes temas tratados en cada capítulo. En este último capítulo, se ofrecerá una visión general de todas estas conclusiones, permitiendo comprender el alcance y la importancia del trabajo realizado. Además, se reflexionará sobre el impacto potencial de estas conclusiones y se propondrán nuevas líneas de investigación que pueden ser de interés en el futuro.

7.1. Introducción

Al inicio de esta tesis, se ha puesto de manifiesto que el consumo de contenido multimedia, en particular el de vídeo, ha adquirido un papel cada vez más importante en la vida de las personas. Es probable que el consumo de este tipo de contenido continúe aumentando en los próximos años gracias a tecnologías emergentes como el vídeo-360, la realidad virtual (VR) y la realidad aumentada (AR), entre otras. Además, debido a la facilidad para generar (participar), compartir y consumir vídeos a través de redes sociales y plataformas de *streaming* —esto se ha convertido en una actividad común para muchos de nosotros, en la generación z como en las generaciones siguientes—. Estas, y otras razones hacen que la preparación de contenido de vídeo sea un área de investigación principal para muchos grupos de investigación. En este contexto, es fundamental desarrollar técnicas y soluciones que permitan ofrecer una experiencia de visualización óptima para el usuario final, procurando reducir el uso de recursos computacionales —fundamental para minimizar los gastos operativos (OpEx) en plataformas de pago por uso—.

En esta tesis, hemos propuesto varias aproximaciones para preparar el contenido de vídeo y que pueda ser transmitido en un escenario de vídeo bajo demanda utilizando el formato DASH, buscando mejorar el compromiso entre calidad de vídeo y uso de recursos computacionales. La codificación/-transcodificación de vídeo está presente desde las primeras etapas en cuanto a la producción de contenido se refiere —desde el plató hasta las salas de pre-producción; generalmente VOD, o desde el teléfono móvil hasta las redes sociales; generalmente vídeo en directo—. En este sentido, focalizados en la generación de contenido para VOD, realizamos un estudio exhaustivo sobre los aportes realizados en la última década por la comunidad científica sobre la codificación/transcodificación de vídeo en la Nube (modelo de consumo de recursos de facto para operaciones computaciones simples o complejas) y de los trabajos relacionados con este tema. La codificación de vídeo basada en la Nube permite a las empresas y organizaciones aprovechar la escalabilidad, flexibilidad y rentabilidad de la Nube para codificar sus contenidos de vídeo. Una de las principales ventajas de utilizar la codificación de vídeo basada en la Nube es la capacidad de ampliar o reducir la escala según sea necesario para satisfacer las demandas cambiantes. En este punto, realizamos un estudio sobre la evolución del *streaming* con especial atención en sus últimos métodos, la importancia de los procesos de codificación, la importancia de los sistemas en la Nube en los entornos multimedia. También, agrupamos los trabajos según el tipo de virtualización y entrega de contenido en la Nube,

y analizamos sus propuestas.

Teniendo en cuenta las soluciones planteadas en la literatura, y por la experiencia previa del equipo de investigación, se propone desarrollar un sistema de codificación que utilice una valor de métrica de calidad de vídeo como parámetro de entrada del sistema de codificación, con el objetivo de ajustar el contenido de toda la secuencia a una calidad deseada. Teniendo esto en consideración, como primera aproximación, se desarrolla un algoritmo que codifica la versión reducida de cada trozo del vídeo (segmento) a varios niveles de calidad, y dado este conjunto de valores se realiza una interpolación para encontrar el valor del parámetro de codificación (CRF) que se Dado que la propuesta inicial demostró ser factible, se planteó un enfoque innovador que introduce un nuevo algoritmo para determinar el valor del CRF para cada segmento del vídeo. En este proceso, la codificación de cada fragmento se itera siguiendo la estrategia de descenso de gradiente. Este método ajusta los parámetros en dirección a la función de pérdida en cada iteración, con un tamaño de paso determinado, hasta alcanzar un punto donde la función de pérdida deja de disminuir significativamente —cabe destacar que el descenso de gradiente es uno de los métodos de optimización más populares y es ampliamente utilizado en el campo del aprendizaje automático—.

Como siguiente paso, y persiguiendo este el mismo objetivo, se realiza una análisis exhaustivo sobre el impacto de utilizar el vídeo a baja escala (*downscaling*) para detectar las escenas de un vídeo respecto a su escala original. En este punto analizamos las repercusiones en tiempo, calidad y tamaño de los vídeos codificados; considerando el uso de recursos. Esto con el objetivo de plantear una mejora en los sistemas de codificación que utilicen la codificación basada en escenas.

Si bien el proceso de investigación de los trabajos antes mencionados se realizó en una arquitectura basada en contenedores —contenedores con Docker sobre una máquina virtual con OpenStack—, no se estaba aprovechando el potencial que esta tecnología proporciona. Por ello, guiados por las nuevas estrategias de consumo y aprovisionamiento de recursos de la Nube, se diseñó e implementó una arquitectura que paralelice los trabajos involucrados en la preparación de contenido, por ejemplo, la codificación/transcodifiación o el cálculo de la calidad del vídeo procesado en sus diferentes representaciones. Tomando en consideración que las plataformas de *streaming* son capaces de generar miles de horas de contenido en tan poco tiempo, buscamos una solución dinámica y elástica para realizar las tareas de codificación. Por tanto, se propuso un sistema de codificación basado en el paradigma FaaS sobre

una plataforma *serverless* como una solución. Se probaron varias tecnologías antes de dar con Knative; plataforma serverless que nos permitía controlar la concurrencia de los trabajo para no saturar a los pods. Nuestra arquitectura fue probada con tres funciones encapsuladas en contenedores del orden de 100 MB, dos para trabajos de codificación (con x264 y AV1) y otra para calcular la calidad del vídeo codificado (con VMAF). De los pipelines implementados, en sus diferentes contextos, se realizó un análisis minucioso del uso de recursos.

A lo largo de esta tesis, hemos observado que, en la mayoría de los estudios propuestos, la fase de experimentación incluye el análisis de la calidad del vídeo resultante como método de validación. Dado que la evaluación de la calidad del vídeo es un aspecto clave en la mayoría de las investigaciones relacionadas con la preparación de contenido de vídeo, nos propusimos desarrollar una herramienta integral que incorpore diversas métricas de calidad.

Así, creamos VQMTK, una herramienta de código abierto que integra 14 métricas en un contenedor Docker, permitiendo una evaluación multiplataforma y simplificando el proceso de análisis. VQMTK cuenta con una interfaz de usuario intuitiva a través de Jupyter notebooks y un script Bash unificador, facilitando su uso en diversos entornos.

En la siguiente subsección mostramos las principales conclusiones de esta tesis siguiendo el orden de los capítulos.

7.2. Conclusiones y Aportaciones

A lo largo de esta tesis doctoral, se han abordado diversos aspectos clave relacionados con la codificación de vídeo en la Nube y la evaluación de la calidad de vídeo en sistemas de transmisión adaptativa. A continuación, se presentan las conclusiones detalladas y amplias de cada capítulo, destacando los hallazgos y contribuciones más importantes en un orden coherente.

Capítulo 2: En este capítulo, se ha realizado un análisis exhaustivo de la codificación de vídeo en la Nube, centrándose en la evolución de la transmisión y la importancia de los sistemas en la Nube en entornos multimedia. La revisión sistemática de la literatura (SLR) permitió identificar y clasificar 49 documentos relevantes según el tipo de tecnología de virtualización y

la entrega de contenido multimedia en la Nube. Además, se ha examinado en detalle la información contenida en sus metadatos. Se han identificado y discutido las principales contribuciones en áreas como la codificación de segmentos de vídeo basada en la calidad, la mejora de la codificación DASH mediante análisis de escenas y técnicas de reducción de escala, la codificación de segmentos con tamaño variable y la codificación de vídeo en la Nube utilizando funciones *serverless*. También se ha presentado un conjunto de herramientas y métricas de calidad de vídeo utilizadas en este campo. Esta revisión proporciona una visión amplia y actualizada del estado del arte en la codificación de vídeo en la Nube y sienta las bases para futuras investigaciones y desarrollos en este ámbito que está en constante evolución.

Capítulo 3: Este capítulo se centra en la importancia de la preparación de contenido en los sistemas HAS y en la maximización de la eficiencia en el proceso de codificación (calidad frente a tamaño y tiempo). Se han presentado dos enfoques de codificación adaptativa basada en la calidad para preparar el contenido de vídeo para transmisión VOD con DASH. En ambos casos, cada segmento de un vídeo sin procesar se codifica para proporcionar una calidad objetivo mediante la resolución de un problema de optimización. Se ha realizado un análisis a baja resolución (240p) para mejorar el rendimiento y se han comparado los enfoques con el uso de un valor de CRF fijo por segmento. Los experimentos demuestran que los enfoques propuestos pueden reducir el tamaño del vídeo codificado en aproximadamente un 10 % en comparación con un CRF fijo por segmento para obtener la misma calidad media.

Capítulo 4: Se ha propuesto una solución de codificación basada en escenas para mejorar la eficiencia del proceso de codificación DASH, logrando un mejor rendimiento en términos de uso de recursos, calidad y tasa de bits. Esta solución implica aplicar un agresivo *downscaling* al vídeo de entrada (de 2160p a 240p) para detectar las escenas y, a continuación, realizar una codificación por segmentos basada en las escenas obtenidas. Los resultados demuestran que el *downscaling* no afecta a la calidad final de los vídeos codificados y ofrece beneficios como la reducción del uso de recursos y tiempos medios totales de codificación (mejora media del 28 % en x264 y del 9 % usando VP9) y la reducción del tamaño de los vídeos resultantes (reducción media del 7 % con x264 y del 5.8 % con VP9).

Capítulo 5: En este capítulo, se ha explorado la aplicación de un sistema de codificación basado en el paradigma FaaS sobre una plataforma *serverless* como solución óptima para preparar contenidos utilizando codificaciones eficientes para plataformas de *streaming*. Se ha desarrollado tres funciones sin servidor basadas en eventos (x264, AV1 y VMAF) y se han encapsulado en tres pequeños contenedores. Se ha analizado su comportamiento en una plataforma sin servidor local con recursos computacionales limitados, evaluando la escalabilidad y el consumo de recursos en distintos estados del contenedor, variando el número máximo de réplicas y los recursos asignados a las mismas (réplicas de función *fat* y *slim*).

Los resultados demuestran el buen rendimiento de las funciones *serverless* propuestas en términos de escalabilidad y distribución de trabajos entre las réplicas. Al utilizar réplicas *slim*, se logra reducir el tiempo total de codificación en un 18 %. Por otro lado, las réplicas *fat* son más adecuadas para escenarios de *streaming* de vídeo en vivo, ya que disminuyen el tiempo de codificación por segmento.

Adicionalmente, se ha evidenciado la capacidad para llevar a cabo una codificación multirresolución de cada segmento a través de una única invocación de función. Esto se complementa con la creación de un flujo de trabajo basado en eventos, específicamente utilizando `CloudEvents`, que permite encadenar múltiples etapas en la preparación de contenido de vídeo.

Este enfoque de *pipeline serverless* proporciona una solución flexible y escalable para manejar tareas complejas de procesamiento de vídeo.

Capítulo 6: Dado el crecimiento sin precedentes del contenido multimedia en Internet, especialmente el vídeo, las plataformas de transmisión deben garantizar un cierto nivel de calidad al preparar su contenido. Para ello, se han desarrollado diversas métricas de calidad de vídeo tanto para uso gratuito como comercial. En este capítulo, se ha creado una herramienta multiplataforma, VQMTK, que integra muchas de estas métricas en una imagen de contenedor Docker, compuesta por una interfaz web y un script Bash que combina 14 métricas de calidad de vídeo y valores SI-TI.

Las pruebas de rendimiento demuestran que `VQMTK` es altamente eficiente y capaz de manejar todas las métricas implementadas utilizando muestras de vídeo en 4K. Esta herramienta tiene aplicaciones en trabajos relacionados con la codificación de vídeo, generación de conjuntos de datos para entrenar Redes Neuronales Profundas y en entornos de aprendizaje.

En resumen, esta tesis doctoral ha abordado aspectos cruciales de la codificación de vídeo en la Nube y la evaluación de la calidad de vídeo, aportando soluciones y herramientas innovadoras que pueden mejorar la eficiencia y la calidad en la preparación y transmisión de contenido multimedia. Los resultados y contribuciones de cada capítulo proporcionan información valiosa y técnica que puede servir de base para futuras investigaciones y desarrollos en este campo que está en constante evolución.

7.3. Desafíos y temas abiertos de investigación relacionados con el trabajo de tesis

La tesis presentó avances significativos en el área de la codificación de vídeo y de las arquitecturas de cómputo en la Nube para realizar estas tareas de codificación.Sin embargo, como en cualquier campo de investigación, aún quedan desafíos y oportunidades para seguir avanzando. En esta sección, se presentan varios temas abiertos y desafíos en áreas como la preparación de contenido de vídeo, la segmentación de vídeo, la codificación basada en la calidad y otros aspectos relacionados con las arquitecturas de cómputo en la Nube y en el borde.

- Mejoras en la preparación de contenido de vídeo: Uno de los desafíos actuales en la transmisión de vídeo es la preparación del contenido para su distribución, especialmente en tiempo real. A medida que los códecs de vídeo evolucionan, surgen nuevas técnicas para optimizar la calidad y el rendimiento de la transmisión. Es importante investigar en métodos avanzados que aprovechen al máximo las capacidades de los códecs modernos y se adapten a las condiciones de la red en tiempo real. Para lograrlo, se pueden utilizar tecnologías como la inteligencia artificial y técnicas como el aprendizaje profundo para ajustar los parámetros precisos en la codificación, preparación y distribución de cada segmento de vídeo. Es necesario considerar la calidad y el estado de la red en todo momento, ya sea en una transmisión en vivo o bajo demanda. De esta forma, se puede optimizar la experiencia del usuario y garantizar una transmisión fluida y de alta calidad.

- Segmentación de vídeo inteligente: La segmentación de vídeo es crucial para la adaptabilidad y la calidad de la transmisión. Investigar en

técnicas de segmentación más avanzadas y adaptables que puedan optimizar la calidad de la transmisión en función del tipo de contenido, las condiciones de la red y las capacidades de los dispositivos de los usuarios.

- Integración de arquitecturas *Edge Computing* y optimización de baja latencia: Actualmente, hay un margen significativo para mejorar en este ámbito. Es importante investigar en nuevas técnicas y herramientas que permitan una integración más efectiva y una optimización de la latencia en cada tramo de la transmisión de vídeo. Esto puede mejorar considerablemente la experiencia del usuario y garantizar una entrega eficiente del contenido de vídeo en tiempo real.

- Explorar el uso de infraestructuras serverless con acceso a aceleradores para mejorar el proceso de codificación de vídeo de muy alta resolución y de vídeo 360.

- Mejora en la gestión de CDN para VOD: Desarrollar mecanismos de descubrimiento de red en infraestructuras de Nube y borde que permitan a las CDN tener una comprensión completa de la red y proporcionar asignaciones óptimas de recursos para reducir la falta de caché al mínimo.

- Empoderamiento nativo de navegadores web con protocolos de *streaming*: Potenciar a los navegadores web con los protocolos de transmisión del futuro, conscientes del estado de la red, la topología, el estado de CDN y otros aspectos, permitiría al sistema predecir la intención del usuario con respecto a la reproducción de vídeo.

- Conciencia de las capacidades de QoS en la transmisión de vídeo: Adaptar los protocolos de transmisión de vídeo para enviar señales directamente a las infraestructuras de Nube y borde, de modo que puedan gestionar dinámicamente los requisitos de QoS de acuerdo con las necesidades del vídeo.

- Uso de segmentos de red en la Nube y el borde: Investigar la introducción de segmentos de red en infraestructuras de Nube y borde, que permitan garantizar la calidad de servicio y los acuerdos de nivel de servicio en la entrega de vídeo.

Estos desafíos y temas abiertos presentan oportunidades emocionantes y posibles trabajos futuros para avanzar en la investigación en el campo de

la transmisión de vídeo y las arquitecturas de cómputo en la Nube y en el borde. Abordar estos problemas o parte de ellos podría conllevar mejoras significativas en la calidad, eficiencia y experiencia del usuario en la transmisión de vídeo en el futuro.

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

Aaron, A., Li, Z., Manohara, M., De Cock, J. y Ronca, D. Per-title encode optimization. *The Netflix Techblog*, 2015.

Abdallah, M., Griwodz, C., Chen, K.-T., Simon, G., Wang, P.-C. y Hsu, C.-H. Delay-sensitive video computing in the cloud: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14(3s), 2018. ISSN 1551-6857.

Adzic, V., Kalva, H. y Furht, B. Optimizing video encoding for adaptive streaming over http. *IEEE Transactions on Consumer Electronics*, vol. 58(2), páginas 397–403, 2012.

Afonso, M., Sole, J., Krasula, L., Li, Z. y Tandon, P. Cambi: Introduction and latest advances. En *Proceedings of the 1st Mile-High Video Conference*, MHV '22, página 105–106. Association for Computing Machinery, New York, NY, USA, 2022. ISBN 9781450392228.

Al-hammouri, M., Madani, B., Aloqaily, M., Ridhawi, I. A. y Jararweh, Y. Scalable video streaming for real-time multimedia applications over dds middleware for future internet architecture. En *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, páginas 1–6. 2018.

Amirpour, H., Çetinkaya, E., Timmerer, C. y Ghanbari, M. Towards optimal multirate encoding for http adaptive streaming. En *MultiMedia Modeling* (editado por J. Lokoč, T. Skopal, K. Schoeffmann, V. Mezaris,

X. Li, S. Vrochidis y I. Patras), páginas 469–480. Springer International Publishing, Cham, 2021. ISBN 978-3-030-67832-6.

ANATOLIY ZABROVSKIY, V. K. R. K. C. T. R. P., PRATEEK AGRAWAL. Fspot: Fast and efficient video encoding workloads over amazon spot instances. *Computers, Materials & Continua*, vol. 71(3), páginas 5677–5697, 2022. ISSN 1546-2226.

ANTON, M., AVINASH, D., DEVANG, S. y MURTHY, P. Production media management: Transforming media workflows by leveraging the cloud. 2021. Disponible en https://t.ly/AeJW (último acceso, Octubre, 2022).

AO, L., IZHIKEVICH, L., VOELKER, G. M. y PORTER, G. Sprocket: A Serverless Video Processing Framework. En *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, páginas 263–274. Association for Computing Machinery, New York, NY, USA, 2018a. ISBN 9781450360111.

AO, L., IZHIKEVICH, L., VOELKER, G. M. y PORTER, G. Sprocket: A serverless video processing framework. En *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, página 263–274. Association for Computing Machinery, New York, NY, USA, 2018b. ISBN 9781450360111.

APPLE. Http live streaming. 2022. Disponible en https://developer.apple.com/streaming/ (último acceso, Noviembre, 2022).

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I. ET AL. A view of cloud computing. *Communications of the ACM*, vol. 53(4), páginas 50–58, 2010.

ASIF, M., MAJEED, S., TAJ, I. A., BIN AHMED, M. y ZIAUDDIN, S. M. Exploiting mb level parallelism in h.264/avc encoder for multi-core platform. En *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, páginas 125–130. 2014.

AZURE. Términos de informática en la nube. 2022. Disponible en https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/ (último acceso, Octubre, 2022).

BACCOUR, E., ERBAD, A., BILAL, K., MOHAMED, A. y GUIZANI, M. PCCP: Proactive Video Chunks Caching and Processing in edge networks. *Future Generation Computer Systems*, vol. 105, páginas 44–60, 2020. ISSN 0167739X.

Banijamali, A., Pakanen, O.-P., Kuvaja, P. y Oivo, M. Software architectures of the convergence of cloud computing and the internet of things: A systematic literature review. *Information and Software Technology*, vol. 122, página 106271, 2020. ISSN 0950-5849.

Barais, O., Bourcier, J., Bromberg, Y.-D. y Dion, C. Towards microservices architecture to transcode videos in the large at low costs. En *2016 International Conference on Telecommunications and Multimedia (TEMU)*, páginas 1–6. IEEE, 2016. ISBN 978-1-4673-8409-4.

Barlas, G. Cluster-based optimized parallel video transcoding. *Parallel Computing*, vol. 38(4-5), páginas 226–244, 2012. ISSN 01678191.

Battista, S., Meardi, G., Ferrara, S., Ciccarelli, L., Maurer, F., Conti, M. y Orcioni, S. Overview of the low complexity enhancement video coding (lcevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, páginas 1–1, 2022.

Benkacem, I., Taleb, T., Bagaa, M. y Flinck, H. Performance benchmark of transcoding as a virtual network function in CDN as a service slicing. En *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2018-April, páginas 1–6. IEEE, 2018. ISBN 978-1-5386-1734-2. ISSN 15253511.

Bentaleb, A., Taani, B., Begen, A. C., Timmerer, C. y Zimmermann, R. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys Tutorials*, vol. 21(1), páginas 562–585, 2019.

Bhat, D., Rizk, A., Zink, M. y Steinmetz, R. Network assisted content distribution for adaptive bitrate video streaming. En *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, página 62–75. Association for Computing Machinery, New York, NY, USA, 2017. ISBN 9781450350020.

Bitmovin. Bitmovin video developer report 2021. Informe técnico, 2021.

Blum, N., Lachapelle, S. y Alvestrand, H. Webrtc: Real-time communication for the open web platform. *Communications of the ACM*, vol. 64(8), páginas 50–54, 2021.

Bross, B., Chen, J., Ohm, J.-R., Sullivan, G. J. y Wang, Y.-K. Developments in international video coding standardization after avc, with

an overview of versatile video coding (vvc). *Proceedings of the IEEE*, vol. 109(9), páginas 1463–1493, 2021a.

BROSS, B., WANG, Y.-K., YE, Y., LIU, S., CHEN, J., SULLIVAN, G. J. y OHM, J.-R. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31(10), páginas 3736–3764, 2021b.

BRUCKSTEIN, A. M., ELAD, M. y KIMMEL, R. Down-scaling for better transform compression. *IEEE Transactions on Image Processing*, vol. 12(9), páginas 1132–1144, 2003.

CHEN, Y., MURHERJEE, D., HAN, J., GRANGE, A., XU, Y., LIU, Z., PARKER, S., CHEN, C., SU, H., JOSHI, U., CHIANG, C.-H., WANG, Y., WILKINS, P., BANKOSKI, J., TRUDEAU, L., EGGE, N., VALIN, J.-M., DAVIES, T., MIDTSKOGEN, S., NORKIN, A. y DE RIVAZ, P. An overview of core coding tools in the av1 video codec. En *2018 Picture Coding Symposium (PCS)*, páginas 41–45. 2018.

CHEN, Y., WU, K. y ZHANG, Q. From qos to qoe: A tutorial on video quality assessment. *IEEE Communications Surveys Tutorials*, vol. 17(2), páginas 1126–1165, 2015.

CHENG, R., WU, W., LOU, Y. y CHEN, Y. A Cloud-Based Transcoding Framework for Real-Time Mobile Video Conferencing System. En *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, páginas 236–245. IEEE, 2014. ISBN 978-1-4799-4425-5.

CHIKKERUR, S., SUNDARAM, V., REISSLEIN, M. y KARAM, L. J. Objective video quality assessment methods: A classification, review, and performance comparison. *IEEE Transactions on Broadcasting*, vol. 57(2), páginas 165–182, 2011.

CHING-CHENG HUANG, JIANN-JONE CHEN y YAO-HONG TSAI. A Dynamic and Complexity Aware cloud scheduling algorithm for video transcoding. En *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, páginas 1–6. IEEE, 2016. ISBN 978-1-5090-1552-8.

CID, E. Cie technical report: Colorimetry. cie pub no. 15.3. 2004.

CISCO. Cisco annual internet report (2018 al 2023). White paper, Cisco Systems, 2020.

CloudEvents. Cloudevents project. https://cloudevents.io/, 2022. Accessed: 2022-08-01.

Covell, M., Arjovsky, M., Lin, Y.-c. y Kokaram, A. Optimizing transcoder quality targets using a neural network with an embedded bitrate model. *Electronic Imaging*, vol. 2016(2), páginas 1–7, 2016.

Daede, T., Norkin, A. y Brailovskiy, I. Video Codec Testing and Quality Measurement. 2017.

Dar, Y. y Bruckstein, A. M. Improving low bit-rate video coding using spatio-temporal down-scaling. *arXiv:1404.4026*, 2014.

DASH-IF. Dash and webrtc-based streaming. 2022. Disponible en https://dashif.org/webRTC/ (último acceso, Octubre, 2022).

De Cock, J., Li, Z., Manohara, M. y Aaron, A. Complexity-based consistent-quality encoding in the cloud. En *2016 IEEE International Conference on Image Processing (ICIP)*, páginas 1484–1488. 2016.

Deriche, M., Beghdadi, A. y Chetouani, A. A distortion-based ranking of image quality indices using mutual information. En *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, páginas 484–487. 2010.

Diaz-Sanchez, D., Marin-Lopez, A., Almenarez, F., Sanchez-Guerrero, R. y Arias, P. A distributed transcoding system for mobile video delivery. En *2012 5th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*, páginas 10–16. IEEE, 2012. ISBN 978-1-4673-2994-1.

Díaz-Sánchez, D., Sánchez-Guerrero, R., Arias, P., Almenarez, F. y Marín, A. A distributed transcoding and content protection system. *Telecommunication Systems*, vol. 61(1), páginas 59–76, 2016. ISSN 1018-4864.

Dong, Y., Zhang, X., Zhao, Y. y Song, L. A containerized media cloud for video transcoding service. En *2018 IEEE International Conference on Consumer Electronics (ICCE)*, páginas 1–4. IEEE, 2018a. ISBN 978-1-5386-3025-9. ISSN 2158-4001.

Dong, Y., Zhang, X., Zhao, Y. y Song, L. A containerized media cloud for video transcoding service. En *2018 IEEE International Conference on Consumer Electronics (ICCE)*, páginas 1–4. 2018b.

DWIVEDI, Y. K., ISMAGILOVA, E., RANA, N. P. y RAMAN, R. Social media adoption, usage and impact in business-to-business (b2b) context: A state-of-the-art literature review. *Information Systems Frontiers*, páginas 1–23, 2021.

EEROLA, T., LENSU, L. T., KÄLVIÄINEN, H. y BOVIK, A. C. Study of no-reference image quality assessment algorithms on printed images. *Journal of Electronic Imaging*, vol. 23(6), páginas 1 – 12, 2014.

EGIAZARIAN, K., ASTOLA, J., PONOMARENKO, N., LUKIN, V., BATTISTI, F. y CARLI, M. New full-reference quality metrics based on hvs. En *Proceedings of the second international workshop on video processing and quality metrics*, vol. 4. 2006.

ESCOLAR, A. M., ALCARAZ-CALERO, J. M., SALVA-GARCIA, P., BERNABE, J. B. y WANG, Q. Adaptive network slicing in multi-tenant 5g iot networks. *IEEE Access*, vol. 9, páginas 14048–14069, 2021.

FAN, Q., YIN, H., MIN, G., YANG, P., LUO, Y., LYU, Y., HUANG, H. y JIAO, L. Video delivery networks: Challenges, solutions and future directions. *Computers Electrical Engineering*, vol. 66, páginas 332–341, 2018. ISSN 0045-7906.

FARHAD, S. M., BAPPI, M. S. I. y GHOSH, A. Dynamic Resource Provisioning for Video Transcoding in IaaS Cloud. En *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, páginas 380–384. IEEE, 2016. ISBN 978-1-5090-4297-5.

FFMPEG. Ffmpeg toll. https://ffmpeg.org/, 2022. Accessed: 2022-08-01.

FOULADI, S., WAHBY, R. S., SHACKLETT, B., BALASUBRAMANIAM, K. V., ZENG, W., BHALERAO, R., SIVARAMAN, A., PORTER, G. y WINSTEIN, K. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads. En *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, páginas 363–376. 2017.

GAO, G. y WEN, Y. Morph: A Fast and Scalable Cloud Transcoding System. En *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, páginas 1160–1163. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450336031.

García-Lucas, D., Cebrián-Márquez, G. y Cuenca, P. Rate-distortion/complexity analysis of hevc, vvc and av1 video codecs. *Multimedia Tools and Applications*, vol. 79(39), páginas 29621–29638, 2020. ISSN 1573-7721.

Garcia-Pineda, M., Felici-Castell, S. y Segura-Garcia, J. Using factor analysis techniques to find out objective video quality metrics for live video streaming over cloud mobile media services. *Netw. Protoc. Algorithms*, vol. 8(1), páginas 126–147, 2016.

Garcia-Pineda, M., Segura-Garcia, J. y Felici-Castell, S. Estimation techniques to measure subjective quality on live video streaming in Cloud Mobile Media services. *Computer Communications*, vol. 118, páginas 27–39, 2018. ISSN 0140-3664.

García-Pineda, M., Felici-Castell, S. y Segura-García, J. Adaptive sdn-based architecture using qoe metrics in live video streaming on cloud mobile media. En *2017 Fourth International Conference on Software Defined Systems (SDS)*, páginas 100–105. 2017.

Grois, D., Nguyen, T. y Marpe, D. Coding efficiency comparison of av1/vp9, h.265/mpeg-hevc, and h.264/mpeg-avc encoders. En *2016 Picture Coding Symposium (PCS)*, páginas 1–5. 2016.

GSM. An introduction to network slicing. Informe técnico, 2017.

Guanyu Gao, Wen, Y. y Westphal, C. Resource provisioning and profit maximization for transcoding in Information Centric Networking. En *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, páginas 97–102. IEEE, 2016. ISBN 978-1-4673-9955-5.

Guo, L., De Cock, J. y Aaron, A. Compression performance comparison of x264, x265, libvpx and aomenc for on-demand adaptive streaming applications. En *2018 Picture Coding Symposium (PCS)*, páginas 26–30. 2018.

Gutiérrez-Aguado, J. Adapting embeded Tomcat to develop event-driven serverless functions. En *JCIS2022*. SISTEDES, 2022.

Gutiérrez-Aguado, J., Peña-Ortiz, R., García-Pineda, M. y Claver, J. Cloud-based elastic architecture for distributed video encoding: Evaluating H.265, VP9, and AV1. *Journal of Network and Computer Applications*, vol. 171, 2020a.

GUTIÉRREZ-AGUADO, J., PEÑA-ORTIZ, R., GARCÍA-PINEDA, M. y CLAVER, J. M. Cloud-based elastic architecture for distributed video encoding: Evaluating H.265, VP9, and AV1. *Journal of Network and Computer Applications*, vol. 171, página 102782, 2020b. ISSN 1084-8045.

GUTIÉRREZ-AGUADO, J., PEÑA-ORTIZ, R., GARCIA-PINEDA, M. y CLAVER, J. M. A cloud-based distributed architecture to accelerate video encoders. *Applied Sciences*, vol. 10(15), 2020a. ISSN 2076-3417.

GUTIÉRREZ-AGUADO, J., PEÑA-ORTIZ, R., GARCIA-PINEDA, M. y CLAVER, J. M. A cloud-based distributed architecture to accelerate video encoders. *Applied Sciences*, vol. 10(15), 2020b. ISSN 2076-3417.

GUTIÉRREZ-AGUADO, J., PEÑA-ORTIZ, R., GARCÍA-PINEDA, M. y CLAVER, J. M. Cloud-based elastic architecture for distributed video encoding: Evaluating H.265, VP9, and AV1. *Journal of Network and Computer Applications*, vol. 171, página 102782, 2020c. ISSN 1084-8045.

HAN, J., LI, B., MUKHERJEE, D., CHIANG, C.-H., GRANGE, A., CHEN, C., SU, H., PARKER, S., DENG, S., JOSHI, U., CHEN, Y., WANG, Y., WILKINS, P., XU, Y. y BANKOSKI, J. A technical overview of av1. *Proceedings of the IEEE*, vol. 109(9), páginas 1435–1462, 2021.

HE, J., WEN, Y., HUANG, J. y WU, D. On the cost–qoe tradeoff for cloud-based video streaming under amazon ec2's pricing models. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24(4), páginas 669–680, 2014.

HEGAZY, M., DIAB, K., SAEEDI, M., IVANOVIC, B., AMER, I., LIU, Y., SINES, G. y HEFEEDA, M. Content-Aware Video Encoding for Cloud Gaming. En *Proceedings of the 10th ACM Multimedia Systems Conference*, MMSys '19, páginas 60–73. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450362979.

HUANG, C.-T., QIN, Z. y KUO, C.-C. J. Multimedia storage security in cloud computing: An overview. En *2011 IEEE 13th International Workshop on Multimedia Signal Processing*, páginas 1–6. 2011.

HUANG, J.-C., WU, C.-Y. y CHEN, J.-J. On high efficient cloud video transcoding. En *2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, páginas 170–173. IEEE, 2015. ISBN 978-1-4673-6499-7.

Huang, T., Zhang, R.-X. y Sun, L. Deep reinforced bitrate ladders for adaptive video streaming. En *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '21, página 66–73. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450384353.

Huang, T.-Y., Johari, R., McKeown, N., Trunnell, M. y Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. En *Proceedings of the 2014 ACM conference on SIGCOMM*, páginas 187–198. 2014.

Huynh-Thu, Q. y Ghanbari, M. The accuracy of PSNR in predicting video quality for different video scenes and frame rates. *Telecommunication Systems*, vol. 49(1), páginas 35–48, 2010.

Huysegems, R., Van Der Hooft, J., Bostoen, T., Rondao Alface, P., Petrangeli, S., Wauters, T. y De Turck, F. Http/2-based methods to improve the live experience of adaptive streaming. En *Proceedings of the 23rd ACM international conference on Multimedia*, páginas 541–550. 2015.

IBM Watson Media Support Center. Internet connection and recommended encoding settings –IBM Watson Media. https://support.video.ibm.com/hc/en-us/articles/207852117-Internet-connection-and-recommended-encoding-settings, ???? Accessed: 2022-07-25.

Ismagilova, E., Dwivedi, Y. K., Slade, E. y Williams, M. D. Electronic word of mouth (ewom) in the marketing context: A state of the art. 2017.

ISO Central Secretary. Information technology —coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s —part 2: Video. Standard ISO/IEC 11172-2:1993, International Organization for Standardization, Geneva, CH, 1993.

ISO Central Secretary. Information technology —dynamic adaptive streaming over http (dash) —part 1: Media presentation description and segment formats. Standard ISO/IEC 23009-1:2022(en), International Organization for Standardization, Geneva, CH, 2022.

ITU-T. Buffer models for media streams on tcp transport. Recommendation G.1022, International Telecommunication Union, 2016a.

ITU-T. Mean opinion score interpretation and reporting. Recommendation P.800.2, International Telecommunication Union, 2016b.

ITU-T. Methods for the subjective assessment of video quality, audio quality and audiovisual quality of internet video and distribution quality television in any environment. Recommendation P.913, International Telecommunication Union, 2021.

ITU-T. Subjective video quality assessment methods for multimedia applications. Recommendation P.910, International Telecommunication Union, 2022.

JANGDA, A., PINCKNEY, D., BRUN, Y. y GUHA, A. Formal foundations of serverless computing. *Proc. ACM Program. Lang.*, vol. 3(OOPSLA), 2019.

JAYASENA, K., LI, L. y XIE, Q. Multi-modal Multimedia Big Data Analyzing Architecture and Resource Allocation on Cloud Platform. *Neurocomputing*, vol. 253, páginas 135–143, 2017. ISSN 09252312.

JIANG, Q., LEE, Y. C. y ZOMAYA, A. Y. Scalable Video Transcoding in Public Clouds. En *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, páginas 70–75. IEEE, 2019. ISBN 978-1-7281-0912-1.

JIANG, X., YU, F. R., SONG, T. y LEUNG, V. C. M. A survey on multi-access edge computing applied to video streaming: Some research issues and challenges. *IEEE Communications Surveys Tutorials*, vol. 23(2), páginas 871–903, 2021.

JIN, Y., WEN, Y. y WESTPHAL, C. Optimal Transcoding and Caching for Adaptive Streaming in Media Cloud: an Analytical Approach. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25(12), páginas 1914–1925, 2015. ISSN 1051-8215.

JOKHIO, F., ASHRAF, A., LAFOND, S., PORRES, I. y LILIUS, J. Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing. En *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, páginas 254–261. IEEE, 2013. ISBN 978-1-4673-5321-2. ISSN 2377-5750.

KALVA, H. The h.264 video coding standard. *IEEE MultiMedia*, vol. 13(4), páginas 86–90, 2006.

Karam, L. J., Ebrahimi, T., Hemami, S. S., Pappas, T. N., Safranek, R. J., Wang, Z. y Watson, A. B. Introduction to the issue on visual media quality assessment. *IEEE Journal of Selected Topics in Signal Processing*, vol. 3(2), páginas 189–192, 2009.

Katsavounidis, I. Dynamic optimizer–a perceptual video encoding optimization framework. *The Netflix Tech Blog*, 2018.

Kerdranvat, M., Chen, Y., Jullian, R., Galpin, F. y François, E. The video codec landscape in 2020. *ITU Journal: ICT Discoveries*, vol. 3(1), páginas 73–83, 2020. ISSN 2616-8375.

Kesavaraja, D. y Shenbagavalli, A. Hadoop scalable Video Transcoding technique in cloud environment. En *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, páginas 1–6. IEEE, 2015. ISBN 978-1-4799-6480-2.

Kesavaraja, D. y Shenbagavalli, A. Framework for Fast and Efficient Cloud Video Transcoding System Using Intelligent Splitter and Hadoop MapReduce. *Wireless Personal Communications*, vol. 102(3), páginas 2117–2132, 2018. ISSN 0929-6212.

Kim, H.-W., Mu, H., Park, J. H., Sangaiah, A. K. y Jeong, Y.-S. Video transcoding scheme of multimedia data-hiding for multiform resources based on intra-cloud. *Journal of Ambient Intelligence and Humanized Computing*, vol. 11(5), páginas 1809–1819, 2020. ISSN 1868-5137.

Kim, M., Han, S., Cui, Y., Lee, H., Cho, H. y Hwang, S. CloudDMSS: robust Hadoop-based multimedia streaming service architecture for a cloud computing environment. *Cluster Computing*, vol. 17(3), páginas 605–628, 2014. ISSN 1386-7857.

Kitchenham, B. y Brereton, P. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, vol. 55(12), páginas 2049–2075, 2013. ISSN 0950-5849.

Kitchenham, B. y Charters, S. Guidelines for performing systematic literature reviews in software engineering. 2007.

Knative. Knative is an open-source enterprise-level solution to build serverless and event driven applications. https://knative.dev/docs/, 2022. Accessed: 2022-08-01.

KUA, J., ARMITAGE, G. y BRANCH, P. A survey of rate adaptation techniques for dynamic adaptive streaming over http. *IEEE Communications Surveys Tutorials*, vol. 19(3), páginas 1842–1866, 2017.

LAI, C.-F., WANG, H., CHAO, H.-C. y NAN, G. A Network and Device Aware QoS Approach for Cloud-Based Mobile Streaming. *IEEE Transactions on Multimedia*, vol. 15(4), páginas 747–757, 2013. ISSN 1520-9210.

LAKSHMAN, T., ORTEGA, A. y REIBMAN, A. Vbr video: tradeoffs and potentials. *Proceedings of the IEEE*, vol. 86(5), páginas 952–973, 1998.

LE DAO, T. H., GIAP, P. V. y XIEM, H. V. Adaptive long-term reference selection for efficient scalable surveillance video coding. En *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, páginas 69–73. 2018.

LEDERER, S. Optimal adaptive streaming formats mpeg-dash & hls segment length. *Bitmovin*, 2015.

LEDERER, S. Optimal adaptive streaming formats mpeg-dash & hls segment length. *Bitmovin*, 2020.

LEDERER, S., MÜLLER, C. y TIMMERER, C. Dynamic adaptive streaming over http dataset. En *Proc. of the 3rd Multimedia Systems Conference (MMSys '12)*, página 89–94. ACM, 2012. ISBN 9781450311311.

LEI, X., JIANG, X. y WANG, C. Design and implementation of streaming media processing software based on rtmp. En *2012 5th International Congress on Image and Signal Processing*, páginas 192–196. 2012.

LI, J., KULKARNI, S. G., RAMAKRISHNAN, K. K. y LI, D. Analyzing open-source serverless platforms: Characteristics and performance. *International Conferences on Software Engineering and Knowledge Engineering*, 2021. ISSN 2325-9000.

LI, X., DARWICH, M., BAYOUMI, M. y SALEHI, M. A. Cloud-based video streaming services: A survey. *arXiv preprint arXiv:2011.14976*, 2020a.

LI, X., SALEHI, M. A., BAYOUMI, M. y BUYYA, R. CVSS: A Cost-Efficient and QoS-Aware Video Streaming Using Cloud Services. En *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, CCGRID '16, páginas 106–115. IEEE, 2016a. ISBN 978-1-5090-2453-7.

Li, X., Salehi, M. A., Bayoumi, M., Tzeng, N.-F. y Buyya, R. Cost-Efficient and Robust On-Demand Video Transcoding Using Heterogeneous Cloud Services. *IEEE Transactions on Parallel and Distributed Systems*, vol. 29(3), páginas 556–571, 2018a. ISSN 1045-9219.

Li, Z., Aaron, A., Katsavounidis, I., Moorthy, A. y Manohara, M. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, vol. 6(2), 2016b.

Li, Z., Bampis, C., Novak, J., Aaron, A., Swanson, K., Moorthy, A. y Cock, J. Vmaf: The journey continues. *Netflix Technology Blog*, vol. 25, 2018b.

Li, Z., Duanmu, Z., Liu, W. y Wang, Z. AVC, HEVC, VP9, AVS2 or AV1? - a comparative study of state-of-the-art video encoders on 4K videos. En *ICIAR*. 2019.

Li, Z., Huang, Y., Liu, G., Wang, F., Zhang, Z. L. y Dai, Y. Cloud transcoder: Bridging the format and resolution gap between Internet videos and mobile devices. En *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video*, páginas 33–38. ACM Press, New York, New York, USA, 2012. ISBN 9781450314305.

Li, Z., Swanson, K., Bampis, C., Krasula, L. y Aaron, A. Toward a better quality metric for the video community. *The Netflix Tech Blog*, página 2, 2020b.

Li, Z.-N., Drew, M. S. y Liu, J. *New Video Coding Standards: H.264 and H.265*, páginas 395–434. Springer International Publishing, Cham, 2014. ISBN 978-3-319-05290-8.

Liu, D., Li, Y., Lin, J., Li, H. y Wu, F. Deep learning-based video coding: A review and a case study. *ACM Comput. Surv.*, vol. 53(1), 2020. ISSN 0360-0300.

Luo, M. R., Cui, G. y Rigg, B. The development of the cie 2000 colour-difference formula: Ciede2000. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur*, vol. 26(5), páginas 340–350, 2001.

MARATHE, N., GANDHI, A. y SHAH, J. M. Docker swarm and kubernetes in cloud computing environment. En *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, páginas 179–184. 2019.

MARTINS, H., ARAUJO, F. y DA CUNHA, P. R. Benchmarking serverless computing platforms. *Journal of Grid Computing*, vol. 18(4), páginas 691–709, 2020. ISSN 1572-9184.

MAURER, F., BATTISTA, S., CICCARELLI, L., MEARDI, G. y FERRARA, S. Overview of mpeg-5 part 2–low complexity enhancement video coding (lcevc). *ITU Journal: ICT Discoveries*, vol. 3(1), 2020.

MELL, P., GRANCE, T. ET AL. The nist definition of cloud computing. 2011.

MERCAT, A., VIITANEN, M. y VANNE, J. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. En *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, página 297–302. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450368452.

MICÓ-ENGUÍDANOS, F., MOINA-RIVERA, W., GUTIÉRREZ-AGUADO, J. y GARCIA-PINEDA, M. Per-title and per-segment crf estimation using dnns for quality-based video coding. *Expert Systems with Applications*, página 120289, 2023. ISSN 0957-4174.

MITTAL, A., MOORTHY, A. K. y BOVIK, A. C. Blind/referenceless image spatial quality evaluator. En *2011 conference record of the forty fifth asilomar conference on signals, systems and computers (ASILOMAR)*, páginas 723–727. IEEE, 2011.

MITTAL, A., MURALIDHAR, G. S., GHOSH, J. y BOVIK, A. C. Blind image quality assessment without human training using latent quality factors. *IEEE Signal Processing Letters*, vol. 19(2), páginas 75–78, 2012.

MITTAL, A., SAAD, M. A. y BOVIK, A. C. A completely blind video integrity oracle. *IEEE Transactions on Image Processing*, vol. 25(1), páginas 289–300, 2016.

MITTAL, A., SOUNDARARAJAN, R. y BOVIK, A. C. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, vol. 20(3), páginas 209–212, 2013.

Moina-Rivera, W., Garcia-Pineda, M., Claver, J. M. y Gutiérrez-Aguado, J. Event-Driven Serverless Pipelines for Video Coding and Quality Metrics. *Journal of Grid Computing*, vol. 21(2), página 20, 2023a. ISSN 1572-9184.

Moina-Rivera, W., Garcia-Pineda, M., Gutiérrez-Aguado, J. y Alcaraz Calero, J. M. Cloud media video encoding: Review and challenges. 2023b.

Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Multi-resolution quality-based video coding system for dash scenarios. En *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '21, página 42–49. Association for Computing Machinery, New York, NY, USA, 2021a. ISBN 9781450384353.

Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Video encoding cloud system for high performance scenarios. En *ACM International Conference on Interactive Media Experiences (IMX) - Doctoral Consortium*. 2020.

Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Codificación de vídeo basada en vmaf para escenarios dash. En *XV Jornadas de Ingeniería Telemática –JITEL 2021*, vol. 15, páginas 116–121. 2021b.

Moina-Rivera, W., Gutiérrez-Aguado, J. y Garcia-Pineda, M. Video quality metrics toolkit: An open source software to assess video quality. *SoftwareX*, vol. 23, página 101427, 2023c. ISSN 2352-7110.

Moina-Rivera, W., Gutiérrez-Aguado, J., Garcia-Pineda, M. y Claver, J. M. Improving dash encoding with scenes and downscaling techniques for vod streaming. En *2021 IEEE Global Communications Conference (GLOBECOM)*, páginas 1–6. 2021c.

Moldovan, A., Ghergulescu, I. y Muntean, C. H. Vqamap: A novel mechanism for mapping objective video quality metrics to subjective mos scale. *IEEE Transactions on Broadcasting*, vol. 62(3), páginas 610–627, 2016.

Moravcik, M. y Kontsek, M. Overview of docker container orchestration tools. En *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, páginas 475–480. 2020.

MUKHERJEE, D., BANKOSKI, J., GRANGE, A., HAN, J., KOLESZAR, J., WILKINS, P., XU, Y. y BULTJE, R. The latest open-source video codec vp9 - an overview and preliminary results. En *2013 Picture Coding Symposium (PCS)*, páginas 390–393. 2013.

MÜLLER, C., LEDERER, S. y TIMMERER, C. An evaluation of dynamic adaptive streaming over http in vehicular environments. En *Proceedings of the 4th Workshop on Mobile Video*, MoVid '12, página 37–42. Association for Computing Machinery, New York, NY, USA, 2012. ISBN 9781450311663.

NANDAKUMAR, D., WU, Y., WEI, H. y TEN-AMI, A. On the accuracy of video quality measurement techniques. En *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, páginas 1–6. 2019.

NETFLIX. Netflix on aws. 2022. Disponible en https://aws.amazon.com/solutions/case-studies/netflix/ (último acceso, Octubre, 2022).

PÄÄKKÖNEN, P., HEIKKINEN, A. y AIHKISALO, T. Online architecture for predicting live video transcoding resources. *Journal of Cloud Computing*, vol. 8(1), páginas 1–24, 2019. ISSN 2192113X.

PANARELLO, A., CELESTI, A., FAZIO, M., PULIAFITO, A. y VILLARI, M. A big video data transcoding service for social media over federated clouds. *Multimedia Tools and Applications*, vol. 79(13-14), páginas 9037–9061, 2020. ISSN 1380-7501.

PANG, Z., SUN, L., HUANG, T., WANG, Z. y YANG, S. Towards QoS-Aware Cloud Live Transcoding: A Deep Reinforcement Learning Approach. En *2019 IEEE International Conference on Multimedia and Expo (ICME)*, páginas 670–675. IEEE, 2019. ISBN 978-1-5386-9552-4. ISSN 1945-788X.

PEREIRA, R., AZAMBUJA, M., BREITMAN, K. y ENDLER, M. An architecture for distributed high performance video processing in the cloud. En *2010 IEEE 3rd International Conference on Cloud Computing*, páginas 482–489. 2010. ISSN 2159-6190.

PEREIRA, R. y TAM, C. Impact of enjoyment on the usage continuance intention of video-on-demand services. *Information Management*, vol. 58(7), página 103501, 2021. ISSN 0378-7206.

PETERSEN, K., FELDT, R., MUJTABA, S. y MATTSSON, M. Systematic mapping studies in software engineering. En *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, páginas 1–10. 2008a.

Petersen, K., Feldt, R., Mujtaba, S. y Mattsson, M. Systematic Mapping Studies in Software Engineering. En *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, páginas 68–77. BCS Learning Development Ltd., Swindon, GBR, 2008b.

Petrangeli, S., Pauwels, D., Van Der Hooft, J., Žiak, M., Slowack, J., Wauters, T. y De Turck, F. A scalable webrtc-based framework for remote video collaboration applications. *Multimedia Tools and Applications*, vol. 78(6), páginas 7419–7452, 2019.

Pinson, M. y Wolf, S. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, vol. 50(3), páginas 312–322, 2004.

Ponomarenko, N., Silvestri, F., Egiazarian, K., Carli, M., Astola, J. y Lukin, V. On between-coefficient contrast masking of dct basis functions. En *Proceedings of the third international workshop on video processing and quality metrics*, vol. 4. Scottsdale USA, 2007.

Pääkkönen, P., Heikkinen, A. y Aihkisalo, T. Architecture for predicting live video transcoding performance on docker containers. En *2018 IEEE International Conference on Services Computing (SCC)*, páginas 65–72. 2018.

Qin, Y., Hao, S., Pattipati, K. R., Qian, F., Sen, S., Wang, B. y Yue, C. Abr streaming of vbr-encoded videos: Characterization, challenges, and solutions. En *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, página 366–378. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 9781450360807.

Qin, Y., Hao, S., Pattipati, K. R., Qian, F., Sen, S., Wang, B. y Yue, C. Quality-aware strategies for optimizing abr video streaming qoe and reducing data usage. MMSys '19, página 189–200. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450362979.

Ran, Y., Shi, Y., Yang, E., Chen, S. y Yang, J. Dynamic resource allocation for video transcoding with QoS guaranteeing in cloud-based DASH system. En *2014 IEEE Globecom Workshops (GC Wkshps)*, páginas 144–149. IEEE, 2014. ISBN 978-1-4799-7470-2.

Ranganathan, P., Stodolsky, D., Calow, J., Dorfman, J., Guevara, M., Smullen IV, C. W., Kuusela, A., Balasubramanian,

R., BHATIA, S., CHAUHAN, P., CHEUNG, A., CHONG, I. S., DASHARATHI, N., FENG, J., FOSCO, B., FOSS, S., GELB, B., GWIN, S. J., HASE, Y., HE, D.-K., HO, C. R., HUFFMAN JR., R. W., INDUPALLI, E., JAYARAM, I., KONGETIRA, P., KYAW, C. M., LAURSEN, A., LI, Y., LOU, F., LUCKE, K. A., MAANINEN, J. P., MACIAS, R., MAHONY, M., MUNDAY, D. A., MUROOR, S., PENUKONDA, N., PERKINS-ARGUETA, E., PERSAUD, D., RAMIREZ, A., RAUTIO, V.-M., RIPLEY, Y., SALEK, A., SEKAR, S., SOKOLOV, S. N., SPRINGER, R., STARK, D., TAN, M., WACHSLER, M. S., WALTON, A. C., WICKERAAD, D. A., WIJAYA, A. y WU, H. K. Warehouse-Scale Video Acceleration: Co-Design and Deployment in the Wild. En *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2021, páginas 600–615. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450383172.

REED, A. y KLIMKOWSKI, B. Leaky streams: Identifying variable bitrate dash videos streamed over encrypted 802.11n connections. En *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, páginas 1107–1112. 2016.

RISCO, S., MOLTÓ, G., NARANJO, D. M. y BLANQUER, I. Serverless workflows for containerised applications in the cloud continuum. *Journal of Grid Computing*, vol. 19(3), página 30, 2021. ISSN 1572-9184.

ROBITZA, W. Understanding rate control modes (x264, x265, vpx). 2017.

RODRÍGUEZ-SÁNCHEZ, R., MARTÍNEZ, J. L., DE COCK, J., SÁNCHEZ, J. L., CLOVER, J. M. y VAN DE WALLE, R. Low delay h. 264/avc bidirectional inter prediction on a gpu. En *2013 IEEE International Conference on Image Processing*, páginas 2111–2115. IEEE, 2013.

SAMETI, S., WANG, M. y KRISHNAMURTHY, D. Stride: Distributed Video Transcoding in Spark. En *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, páginas 1–8. IEEE, 2018a. ISBN 978-1-5386-6808-5. ISSN 2374-9628.

SAMETI, S., WANG, M. y KRISHNAMURTHY, D. Stride: Distributed video transcoding in spark. En *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, páginas 1–8. 2018b.

SAMETI, S., WANG, M. y KRISHNAMURTHY, D. Contrast: Container-based transcoding for interactive video streaming. En *NOMS 2020 - 2020*

IEEE/IFIP Network Operations and Management Symposium, páginas 1–9. 2020.

SANDVINE. 2020 covid internet phenomena spotlight report. Report, Sandvine Incorporated, 2020.

SCHULZRINNE, H., CASNER, S., FREDERICK, R. y JACOBSON, V. Rfc 3550: Rtp: A transport protocol for real-time applications. 2003.

SCHULZRINNE, H., RAO, A., LANPHIER, R., WESTERLUND, M. y STIEMERLING, M. Rfc 7826: Real-time streaming protocol version 2.0. 2016.

SCHWARZMANN, S., HAINKE, N., ZINNER, T., SIEBER, C., ROBITZA, W. y RAAKE, A. Comparing fixed and variable segment durations for adaptive video streaming: A holistic analysis. En *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, página 38–53. Association for Computing Machinery, New York, NY, USA, 2020a. ISBN 9781450368452.

SCHWARZMANN, S., HAINKE, N., ZINNER, T., SIEBER, C., ROBITZA, W. y RAAKE, A. Comparing fixed and variable segment durations for adaptive video streaming: A holistic analysis. En *Proc. of the 11th ACM Multimedia Systems Conference, MMSys '20*, página 38–53. ACM, 2020b. ISBN 9781450368452.

SCHWARZMANN, S., ZINNER, T., GEISSLER, S. y SIEBER, C. Evaluation of the benefits of variable segment durations for adaptive streaming. En *2018 Tenth Int. Conf. on Quality of Multimedia Experience (QoMEX)*, páginas 1–6. 2018.

SEGURA-GARCIA, J., FELICI-CASTELL, S., GARCIA-PINEDA, M., CHIRIVELLA-PEREZ, E. y GUTIERREZ-AGUADO, J. An objective video quality metric for live video streaming over cloud systems: full and non reference approaches. *Procedings XII Jornadas de Ingenieria Telematica (JITEL)*, vol. 8(1), páginas 138–144, 2015.

SEMSARZADEH, M., YASSINE, A. y SHIRMOHAMMADI, S. Video Encoding Acceleration in Cloud Gaming. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25(12), páginas 1975–1987, 2015. ISSN 1051-8215.

SEUFERT, M., EGGER, S., SLANINA, M., ZINNER, T., HOSSFELD, T. y TRAN-GIA, P. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, vol. 17(1), páginas 469–492, 2014.

SEUFERT, M., EGGER, S., SLANINA, M., ZINNER, T., HOSSFELD, T. y TRAN-GIA, P. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys Tutorials*, vol. 17(1), páginas 469–492, 2015.

SHARMA, P., CHAUFOURNIER, L., SHENOY, P. y TAY, Y. C. Containers and virtual machines at scale: A comparative study. En *Proceedings of the 17th International Middleware Conference*, Middleware '16. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450343008.

SHEIKH, H. y BOVIK, A. Image information and visual quality. *IEEE Transactions on Image Processing*, vol. 15(2), páginas 430–444, 2006. ISSN 1941-0042.

SINNO, Z. y BOVIK, A. C. Large-scale study of perceptual video quality. *IEEE Transactions on Image Processing*, vol. 28(2), páginas 612–627, 2019.

SONG LIN, XINFENG ZHANG, QIN YU, HONGGANG QI y SIWEI MA. Parallelizing video transcoding with load balancing on cloud computing. En *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, páginas 2864–2867. IEEE, 2013. ISBN 978-1-4673-5762-3. ISSN 2158-1525.

SOUNDARARAJAN, R. y BOVIK, A. C. Video quality assessment by reduced reference spatio-temporal entropic differencing. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23(4), páginas 684–694, 2013.

STATISTA. Internet and social media users in the world 2022 | statista. 2022. Disponible en https://www.statista.com/statistics/617136/digital-population-worldwide (último acceso, Septiembre, 2022).

SUCIU, G., ANWAR, M. y MIHALCIOIU, R. Virtualized video and cloud computing for efficient elearning. 2017. Copyright - Copyright Çarol I"National Defence University 2017; Características del documento - ; Diagrams; Última actualización - 2021-06-22.

SULLIVAN, G. J., OHM, J., HAN, W. y WIEGAND, T. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22(12), páginas 1649–1668, 2012.

SYAHBANA, Y. A., HERMAN, RAHMAN, A. A. y BAKAR, K. A. Aligned-psnr (apsnr) for objective video quality measurement (vqm) in video

stream over wireless and mobile network. En *2011 World Congress on Information and Communication Technologies*, páginas 330–335. 2011.

TAIBI, D., SPILLNER, J. y WAWRUCH, K. Serverless computing-where are we now, and where are we heading? *IEEE Software*, vol. 38(1), páginas 25–31, 2021.

TALEB, T., FRANGOUDIS, P. A., BENKACEM, I. y KSENTINI, A. CDN Slicing over a Multi-Domain Edge Cloud. *IEEE Transactions on Mobile Computing*, vol. 19(9), páginas 2010–2027, 2020. ISSN 1536-1233.

TANDON, P., AFONSO, M., SOLE, J. y KRASULA, L. Cambi: Contrast-aware multiscale banding index. En *2021 Picture Coding Symposium (PCS)*, páginas 1–5. 2021.

THANG, T. C., HO, Q.-D., KANG, J. W. y PHAM, A. T. Adaptive streaming of audiovisual content using mpeg dash. *IEEE Transactions on Consumer Electronics*, vol. 58(1), páginas 78–85, 2012.

TOSHNIWAL, A., RATHORE, K. S., DUBEY, A., DHASAL, P. y MAHESHWARI, R. Media streaming in cloud with special reference to amazon web services: A comprehensive review. En *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, páginas 368–372. 2020.

TSELIOS, C. y TSOLIS, G. A survey on software tools and architectures for deploying multimedia-aware cloud applications. En *Algorithmic Aspects of Cloud Computing* (editado por I. Karydis, S. Sioutas, P. Triantafillou y D. Tsoumakos), páginas 168–180. Springer International Publishing, Cham, 2016. ISBN 978-3-319-29919-8.

TSENG, H.-W., YANG, T.-T., YANG, K.-C. y CHEN, P.-S. An energy efficient vm management scheme with power-law characteristic in video streaming data centers. *IEEE Transactions on Parallel and Distributed Systems*, vol. 29(2), páginas 297–311, 2018.

VAN LATUM, F., VAN SOLINGEN, R., OIVO, M., HOISL, B., ROMBACH, D. y RUHE, G. Adopting gqm based measurement in an industrial environment. *IEEE Software*, vol. 15(1), páginas 78–86, 1998. ISSN 1937-4194.

VAN MA, L., PARK, J., NAM, J., JANG, J. y KIM, J. An efficient scheduling multimedia transcoding method for DASH streaming in cloud environment. *Cluster Computing*, vol. 22(S1), páginas 1043–1053, 2019. ISSN 15737543.

VAN SOLINGEN, R. y BERGHOUT, E. W. *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

VRANJEŠ, M., BAJČINOVCI, V., GRBIĆ, R. y VAJAK, D. No-reference artifacts measurements based video quality metric. *Signal Processing: Image Communication*, vol. 78, páginas 345–358, 2019. ISSN 0923-5965.

WANG, L., LI, M., ZHANG, Y., RISTENPART, T. y SWIFT, M. Peeking behind the curtains of serverless platforms. En *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, páginas 133–146. USENIX Association, Boston, MA, 2018. ISBN ISBN 978-1-939133-01-4.

WANG, T., PERVEZ, A. y ZOU, H. Vqm-based qos/qoe mapping for streaming video. En *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, páginas 807–812. 2010.

WANG, Y., CHEN, W.-T., WU, H., KOKARAM, A. y SCHAEFFER, J. A cloud-based large-scale distributed video analysis system. En *2016 IEEE International Conference on Image Processing (ICIP)*, páginas 1499–1503. IEEE, 2016. ISBN 978-1-4673-9961-6. ISSN 2381-8549.

WANG, Z., BOVIK, A., SHEIKH, H. y SIMONCELLI, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, vol. 13(4), páginas 600–612, 2004.

WEI, L., CAI, J., FOH, C. H. y HE, B. QoS-Aware Resource Allocation for Video Transcoding in Clouds. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27(1), páginas 49–61, 2017. ISSN 1051-8215.

WEN, Y., ZHU, X., RODRIGUES, J. J. P. C. y CHEN, C. W. Cloud mobile media: Reflections and outlook. *IEEE Transactions on Multimedia*, vol. 16(4), páginas 885–902, 2014.

WICHTLHUBER, M., WICKLEIN, G., WILK, S., EFFELSBERG, W. y HAUSHEER, D. Rt-vqm: Real-time video quality assessment for adaptive video streaming using gpus. En *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16. Association for Computing Machinery, New York, NY, USA, 2016. ISBN 9781450342971.

WIKIPEDIA (Búfer). Entrada: "Búfer de datos". Disponible en https://es.wikipedia.org/wiki/Búfer_de_datos (último acceso, Octubre, 2022).

Wikipedia (Códec de video). Entrada: "Códec de video". Disponible en https://es.wikipedia.org/wiki/Códec_de_video (último acceso, Septiembre, 2022).

Wikipedia (Ubicuo). Entrada: "Ubicuo". Disponible en https://es.wikipedia.org/wiki/Ubicuo (último acceso, Septiembre, 2022).

Wohlin, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. En *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 9781450324762.

Wu, H. R. *Introduction: State of the Play and Challenges of Visual Quality Assessment*, páginas 1–30. Springer International Publishing, Cham, 2015. ISBN 978-3-319-10368-6.

Wu, J., Cheng, B., Yang, Y., Wang, M. y Chen, J. Delay-Aware Quality Optimization in Cloud-Assisted Video Streaming System. *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14(1), 2017. ISSN 1551-6857.

Xing, H., Zhou, Z., Wang, J., Shen, H., He, D. y Li, F. Predicting rate control target through a learning based content adaptive model. En *2019 Picture Coding Symposium (PCS)*, páginas 1–5. 2019.

Xu, Y. y Mao, S. A survey of mobile cloud computing for rich media applications. *IEEE Wireless Communications*, vol. 20(3), páginas 46–53, 2013.

Yang, J., He, S., Lin, Y. y Lv, Z. Multimedia cloud transmission and storage system based on internet of things. *Multimedia Tools and Applications*, vol. 76(17), páginas 17735–17750, 2017. ISSN 1573-7721.

Yang, M., Cai, J., Zhang, W., Wen, Y. y Foh, C. H. Adaptive configuration of cloud video transcoding. En *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2015-July, páginas 1658–1661. IEEE, 2015. ISBN 978-1-4799-8391-9. ISSN 02714310.

Yang, Y., Ming, J. y Yu, N. Color image quality assessment based on ciede2000. *Advances in Multimedia*, vol. 2012, 2012.

Zach, O. y Slanina, M. Content aware segment length optimization for adaptive streaming over http. *Radioengineering*, vol. 27(3), página 8, 2018a.

ZACH, O. y SLANINA, M. Content aware segment length optimization for adaptive streaming over http. *Radioengineering*, vol. 27(3), páginas 819–826, 2018b.

ZAKERINASAB, M. R. y WANG, M. Dependency-aware distributed video transcoding in the cloud. En *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, vol. 26-29-Octo, páginas 245–252. IEEE, 2015. ISBN 978-1-4673-6770-7.

ZHAO, X., ZHANG, S. y DOU, W. Multi-Request Scheduling and Collaborative Service Processing for DASH-Video Optimization in Cloud-Edge Network. En *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, páginas 582–589. IEEE, 2020. ISBN 978-1-7281-8780-8. ISSN 2159-6190.

ZHENG, L., TIAN, L. y WU, Y. A rate control scheme for distributed high performance video encoding in cloud. En *2011 International Conference on Cloud and Service Computing*, páginas 131–133. IEEE, 2011. ISBN 978-1-4577-1637-9.

ZHU, W., LUO, C., WANG, J. y LI, S. Multimedia cloud computing. *IEEE Signal Processing Magazine*, vol. 28(3), páginas 59–69, 2011.

ZHUANG, Z. y GUO, C. Building cloud-ready video transcoding system for Content Delivery Networks (CDNs). En *2012 IEEE Global Communications Conference (GLOBECOM)*, páginas 2048–2053. IEEE, 2012. ISBN 978-1-4673-0921-9.

Al inicio del todo, cual Dulcinea del Toboso:

–Trata de focusarte en las cosas que tienes que hacer [...] fuerza!!! :)–

Alexandra Marinoiu

Al final del camino, cual Don Quijote:

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –

Bien podrán los encantadores quitarme la ventura,

pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero

Don Quijote de la Mancha

Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.

–No es menester firmarla – dijo Don Quijote–,

sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero

Don Quijote de la Mancha

Miguel de Cervantes