

Evaluación de dos algoritmos para distribución concurrente de carga en el códec H265/HEVC usando video 4K

A. Rodríguez-León^{1*}, M. Pérez Malumbres², C.J. Genis Triana¹, G.E. Ovando Chacon³, A. Rodríguez-Gonzalez¹

¹*Departamento de Sistemas y Computación, Tecnológico Nacional de México/I.T. Veracruz, M.A. de Quevedo 2779, Col. Formando Hogar, C.P. 91860, Veracruz, Ver., México*

²*Departamento de Física y Arquitectura de Computadores. Universidad Miguel Hernández, Avd. Universidad s/n 03202, Elche, Comunidad Valenciana, España*

³*Departamento de Ingeniería Mecánica, Tecnológico Nacional de México/I.T. Veracruz, M.A. de Quevedo 2779, Col. Formando Hogar, C.P. 91860, Veracruz, Ver., México*

*abelardo.rl@veracruz.tecnm.mx

Área de participación: Sistemas Computacionales

Resumen

Un códec de video es un software que se basa en una serie de algoritmos para codificar/decodificar (comprimir/descomprimir) un flujo de video. Los códec reducen el tamaño del video eliminando la información redundante, sin que se perciba la pérdida, conservando adecuadamente la calidad. Sin embargo los codificadores de video que trabajan con resoluciones 4K, necesitan un alto nivel de cómputo.

Las técnicas de programación concurrente aprovechan el poder de cómputo de un clúster de alto rendimiento (HPC por sus siglas en inglés) para aplicaciones que tienen requerimientos de cómputo muy altos, reduciendo tiempos de ejecución.

El presente proyecto describe una comparativa entre dos algoritmos concurrentes para codificar, video 4K (3840x20160) usando el códec H265/HEVC que es uno de los estándares que se usan actualmente en codificación comercial. El resultado aportaría una base algorítmica para la posible aplicación en circuitos impresos. Dichos circuitos normalmente están embebidos en equipos como videocámaras o teléfonos celulares

Palabras clave: H265, 4K/UHD, códec paralelo

Abstract

A video codec is a software based in a serie of algorithms for coding/decoding (compress/uncompress) a video stream. The codecs reduce the size of the video by deleting redundant information without loss being perceived, while preserving adequately quality. Nevertheless, the video codecs that work with 4k resolutions, need a high level of compute.

The techniques of concurrent programming use the compute power of a high performance cluster (HPC) for applications that have high requirements of compute, reducing time of executions.

The present project describe a comparative between two concurrent algorithms for coding 4k video (3840x20160) using the H265/HEVC codec which is one of the standards that are using currently in commercial codification. The result would contribute one algorithmic base for the possible application in printed circuits. This circuits normally are embedded in devices as video cameras or cellphones.

Key words: H265, 4K/UHD, parallel codec

Introducción

El video de alta definición (HD) según [1] es un formato de video con una mayor resolución que la definición estándar, que alcanza resoluciones de 1280x720 y 1920x1080 píxeles. Estas resoluciones son las más utilizadas incluso en los dispositivos actuales. Sin embargo en tiempos recientes han empezado a surgir

dispositivos (Cámaras, TV y Smartphone) que manejan imágenes con mayores resoluciones como 4K (3820x2160) y 8K (7680x4320) [2].

Un elemento importante en el manejo del video en dispositivos digitales es el códec. De forma genérica un códec es un software que se basa de un conjunto de complejos algoritmos que pueden codificar y decodificar un flujo de video para su posterior reproducción y manipulación. En términos simples la codificación es la compresión de video y la decodificación es la descompresión del mismo. Un códec al procesar video lleva a cabo una secuencia específica de pasos. El primer paso es la preparación llamada pre-procesamiento, posteriormente la compresión, y por último se envía por algún canal o es almacenado, dependiendo el uso que se le vaya a dar. Posteriormente (aunque no necesariamente de forma inmediata) se decodifica el flujo y finalmente tiene lugar un post-procesamiento, que puede incluir el manejo de errores, la reproducción del video u otras operaciones. De estas dos tareas (codificación y decodificación) nos interesa específicamente la primera que es la que más recursos computacionales ocupa, ya que la decodificación suele ser muy rápida.

El objetivo de la compresión en un códec es reducir el espacio de almacenamiento de un video con respecto a su original (conocido como video crudo), alterando lo menos posible la calidad de la imagen, para que pueda ser almacenada en un minino espacio. Para conseguirlo, aprovechan que las secuencias de video (formadas por fotogramas) tienen información redundante. Esta redundancia se presenta en la dimensión espacial (en un mismo fotograma) y en la dimensión temporal (entre varios fotogramas). Por consiguiente, al eliminar o reducir dicha información redundante se consigue reducir considerablemente el tamaño del video.

Hay varios estándares disponibles para codificar video. Unos de los que más usados en la actualidad es el que han desarrollado por los grupos ISO/IEC Moving Picture Experts Group (MPEG) y el ITU-T Video Coding Experts Group (VCEG). Dichos grupos crearon hace algunos años el H264/AVC [3] y más recientemente H265/HEVC [4]. El H265/HEVC es la evolución del H264/AVC mejorando características importantes como la calidad y tolerancia a fallos en medios de transmisión propensos a errores. El H.265/HEVC ha sido desarrollado pensando en codificar videos de mayor resolución sin menoscabo de calidad, y en un espacio de almacenamiento equivalente o menor a los de su antecesor, pero con un consumo de cómputo hasta un 70% más alto que el H264 [6]. Esto tiene como consecuencia que el tiempo requerido para codificar por ejemplo video 4K sea muy elevado.

Debido a que el H265 tiene un requerimiento de cómputo considerablemente alto [5], si se desea un tiempo de codificación más aceptable se necesita de mucho poder de cómputo al codificar el video. Una alternativa es usar un clúster de computadoras, que contribuya a bajar el tiempo de codificación y así tratar de alcanzar la meta ideal de codificación en tiempo real. La codificación en tiempo real se da cuando el tiempo de codificación de un video es igual o menor a la duración del mismo. Así que si por ejemplo se requiere codificar un video de 5 minutos, el tiempo necesario para codificarlo no deberá ser menor o igual a 5 minutos. Este requerimiento permite que algunas aplicaciones puedan recibir video crudo (de un dispositivo como una cámara), comprimirlo y transmitirlo con un retraso mínimo y constante. La mayoría de los dispositivos actuales que capturan y transmiten video en tiempo real, tienen una limitada capacidad de procesamiento, por lo que sacrifican la calidad de la imagen, generando un flujo de bytes con gran tamaño. Si se sacrifica la calidad de la imagen el resultado obvio es una imagen poco clara e incluso no entendible. Si se sacrifica el tamaño resulta difícil transmitir por el alto requerimiento del canal de comunicaciones.

En base a lo anterior el objetivo de este trabajo fue desarrollar y evaluar dos algoritmos paralelos para codificación de video en clústers de computadoras, usando el códec H265 HEVC con video en resolución 4K. Para lograrlo fue necesario aplicar un paradigma de programación concurrente, específicamente el que se le conoce como paralelismo de datos. Fue posible aplicar el paralelismo de datos en la codificación gracias a que esta disponible del código fuente del códec H.265/HEVC, lo que permitió controlar la forma en que se distribuye el trabajo de codificación entre varios núcleos (cores) a la vez. La anterior división y distribución de trabajos reduce sustancialmente el tiempo de codificación del video. El trabajo fino en este sentido va enfocado en la aplicación de los dos algoritmos de distribución de carga, ya que no todos los algoritmos de distribución dan buenos resultados con cualquier tipo problema.

Estado del arte

En esta sección se analizan otros trabajos publicados, relacionados con el presente trabajo. Los trabajos analizados son respecto a mejoras y aplicaciones paralelas del códec H.264/AVC y H265/HEVC.

Paralelización del codificador H.264 con estimación de movimiento adaptativa en clústers de Pcs

En este trabajo [8], se busca disminuir el coste computacional realizando de manera adaptativa la estimación de movimiento para la codificación con el H.264/AVC en base a una función que maximice la relación entre el rendimiento del codificador y el tiempo de codificación.

El paralelismo se consigue descomponiendo la secuencia de video en bloques de 15 fotogramas consecutivos (GOPs por sus siglas en inglés), de forma que cada bloque sigue un patrón de codificación del tipo IBBPBBP...PBBI. La codificación de cada GOP es una tarea independiente, por lo que los nodos no requieren comunicarse con los demás para codificar su GOP.

Existe una escasa variación del tiempo de codificación a nivel de GOP que permite obtener algoritmos paralelos con buen equilibrio de la carga, sin embargo, al utilizar el mecanismo de estimación de movimiento adaptativa, se introduce un desbalanceo, haciendo que existan diferencias significativas de coste computacional entre los GOPs de una secuencia de video. En este caso, el esquema de asignación de GOPs a procesadores puede jugar un papel importante en cuanto al rendimiento del codificador paralelo se refiere. Para poder estudiar este fenómeno utilizaron dos esquemas de asignación de GOPs: asignación por demanda y asignación por estimación de coste.

La evaluación de los esquemas propuestos se realizó sobre dos videos: Foreman (CIF) y Stockholm (HDTV). Las pruebas se efectuaron sobre un clúster de cuatro nodos con dos núcleos Opteron 246 a 2GHz, 1 GByte de memoria, interconectados por una red Gigabit Ethernet con dos enlaces por nodo y un switch 3Com. El sistema operativo es Linux SuSE 9.1 y el entorno MPI es MPICH-2.

Los resultados obtenidos muestran como los codificadores H264 paralelos, con estimación de movimiento adaptativa, reducen el tiempo de codificación. En la asignación de GOPs por estimación de coste las mejoras están entorno a un 10%, siempre por encima de la asignación por demanda cuando se tienen menos de 8 GOPs por procesador.

En una mejora del mismo trabajo [9] en cuanto a la aceleración (speedup), se dió para el caso de la asignación de GOPs por estimación de coste. Aunque la magnitud de la mejora, en las secuencias consideradas, es relativamente pequeña.

Desarrollo y evaluación de algoritmos paralelos para la compresión de secuencias de video

En este trabajo [10], se desarrollaron 3 versiones paralelas (preasignación, demanda y predictivo) para el códec H264/AVC versión 7.6. Se consideró que la mejor distribución de GOPs fue por Demanda, ya que a pesar de contar con un proceso dedicado para la distribución de GOPs, tenía mejor eficiencia con respecto a la distribución por preasignación. Para el caso de la versión predictiva se necesita disponer de una porción considerable del flujo de video para su buen funcionamiento, por lo que sólo resulta útil para la codificación fuera de línea de video almacenado [11]. En forma resumida podemos decir que las 3 versiones presentan una excelente escalabilidad y eficiencia y que la tasa de bits es la misma en las 3 por lo que la diferencia estriba solo en la latencia y el equilibrio de la carga. En cuanto a la latencia, va creciendo respectivamente para preasignación, demanda y predictivo, lo cual no es bueno. En cuanto al equilibrio de la carga es medianamente bueno para preasignación y bueno para demanda y predictivo.

En base al proyecto anterior se consideró adecuado y como buen punto de partida usar las versiones paralelas del codificador por preasignación y por demanda. Solo hay que ajustar el patrón de codificación ya que para la versión paralela del códec H264/AVC versión 7.6 utilizan el patrón de codificación IBBPBBPBBPBB y los patrones que se usaron en el códec H265/HEVC versión 16.20 aplicados en este proyecto son IIIIIIIIIII y IBBBIBBBIBBB.

Enfoque paralelo con memoria distribuida para el códec HEVC

En un trabajo más reciente [12] proponen varios enfoques de paralelización con el codificador H265/HEVC, usando una plataforma de hardware con memoria distribuida. Hacen una paralelización de grano grueso también a nivel grupo de imágenes (GOP). El enfoque así mismo es la codificación simultánea de varios GOP. Para ello presentan varias versiones de organización de GOPs y distribución de los mismos. Usan dos videos de prueba clase B y E con resoluciones 1920 × 1080 and 1280 × 720.

Las variantes de armado de GOPs que presentan son las siguientes:

DMG-1G: todos los procesos codifican el primer fotograma I y lo incluyen en su lista de imágenes de referencia. Cuando un proceso ha completado su trabajo, le solicita al coordinador que codifique el próximo GOP.

DMG-1B: la secuencia de video se divide en tantas partes como la cantidad de procesadores disponibles. Cada procesador calcula su secuencia de video parcial como una secuencia de video independiente.

DMG-GB: primero, todos los procesos codifican la primera trama I (como lo hace DMG-1G) y luego, cuando un proceso ha completado su trabajo, el proceso coordinador asigna un número fijo de GOPs contiguos conforme los procesos van terminando la tarea inicial.

DMG-AI: siguiendo el esquema DMG-1G, el proceso coordinador asigna un fotograma a cada proceso que queda inactivo (cada fotograma de la secuencia se codifica como un fotograma I).

Los resultados que obtienen son buenos pero están sujetos a que el video completo esté disponible en la mayoría de las variantes de distribución de GOPs. Además las resoluciones de prueba no son 4K. En nuestro trabajo trabajamos todos los videos en 4K y no asumimos de antemano que se tiene toda la secuencia, lo cual hace apto nuestra propuesta para aplicarse por ejemplo en transmisión de video en vivo.

Metodología

Para implementar las versiones paralelas del códec H265 se trabajó directamente con código fuente del códec JCTV-HM versión 16.20. Dicho códec está escrito en lenguaje C++ y está basado en el High Efficiency Video Coding (HEVC) bajo la norma ISO/IEC CD 23008-2 High Efficiency Video Coding.

Toda la implementación del componente paralelo se desarrolló con la librería estándar MPI (Message Passage Interface). El componente paralelo está compuesto por dos algoritmos planteados en los objetivos (preasignación y demanda). Los algoritmos se basan en diferentes formas de distribuir la carga de codificación entre los núcleos del procesador en un sistema de cómputo SMP sobre el cual se ejecuta la versión paralela. La distribución de la carga se tiene que realizar sin afectar el resultado final que se obtiene en la versión secuencial. A continuación se describen los algoritmos de distribución de la carga.

Preasignación:

Se diseñó una estrategia para que cada proceso sepa, en base a su identificador, qué GOPs ha de codificar, repartiéndolos de la forma más equitativa posible. En principio el costo computacional de la codificación de los fotogramas que componen un GOP puede ser variable e impredecible. Esto plantea la conveniencia de hacer una distinción entre un GOP y otro, para hacer un reparto equitativo del trabajo. Con el fin de simplificar la idea de distribución asumiremos un costo computacional similar para cada GOP. Con esta suposición de partida, hacemos el siguiente esquema de distribución: La secuencia de vídeo se divide en GOPs, cada uno de los cuales contendrá una secuencia de fotogramas. El primer fotograma a codificar siempre debe ser de tipo I, para hacer independiente la codificación de GOPs.

Los procesadores son identificados con un índice entre 0 y el número de procesadores disponibles menos 1. Se divide el número de GOPs entre el número de procesadores, y se asigna a cada procesador el número de GOPs consecutivos resultante, de manera cíclica. En caso de que el número de GOPs no sea múltiplo del número de procesadores, entonces el número de GOPs restante n , se asignarán a los n primeros procesadores, a razón de un GOP por procesador. El reparto se define con la siguiente ecuación:

$$n_gops_Pk = \begin{cases} n_gops_as + 1 & k < n_gops_not \\ n_gops_as & k \geq n_gops_not \end{cases}$$

Dónde:

$n_fotogramas$ será el número de fotograma s en la secuencia completa

$n_fotogramas_gop$ es el número de fotograma s que componen un GOP

n_gops es número total de GOPs en la secuencia de vídeo
 $n_gops = n_fotograma\ s / n_fotograma\ s_gop$

p es el número total de procesadores disponibles

n_gops_not es el número de GOPs no asignados
 $n_gops_not = \text{mod}(n_gops, p)$

• n_gops_as es el número de GOPs asignados a cada procesador
 $n_gops_as = (n_gops - n_gops_not) / p$

• n_gops_Pk es el total de GOPs asignados al procesador P_k , donde k esta entre 0 y $p-1$ ($k = 0 : p-1$)

De esta forma cada procesador sabe cuáles son los GOPs consecutivos que le tocan codificar.

La figura 1 muestra esquemáticamente la funcionalidad de la versión de preasignación del códec.

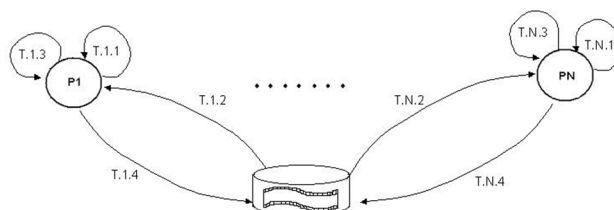


Figura 1. Distribución de carga (GOPs) por Preasignación

Los estados del grafo en la figura 1 son descritos a continuación.

T.1.1. P1 calcula GOP a codificar	T.N.1. PN calcula GOP a codificar
T.1.2. P1 lee GOP del disco	T.N.2. PN lee GOP del disco
T.1.3. P1 codifica GOP	T.N.3. PN codifica GOP
T.1.4. P1 escribe GOP codificado	T.N.4. PN escribe GOP codificado

Una vez terminada la codificación de todos los GOPs de la secuencia, el proceso 0 espera que los demás nodos le envíen un mensaje de terminación para proceder con la unión de todos los GOPs codificados en un solo archivo.

Demanda

El algoritmo por demanda es un esquema más elaborado que permite mejorar el equilibrio de la carga entre los procesadores. Esto se hizo porque en general el número de procesadores no es múltiplo exacto del número de GOPs, lo que ocasiona desbalance de carga al final de la codificación, ya que algunos procesos terminan antes que otros. El resultado de esta espera es que la eficiencia baja sustancialmente en estos casos, sobre todo con secuencias de vídeo de corta duración. Para hacer menos significativas las diferencias de tiempo final, se define un nodo distribuidor y los demás solo codifican. El nodo distribuidor se encarga de indicarle a los demás procesadores que GOPs deben codificar.

En la primera ronda de reparto la asignación es secuencial. Pero después se va asignando los GOPs conforme son solicitados por los nodos codificadores (bajo demanda). Los nodos codificadores reciben la indicación del número de GOP a codificar, lo codifican y cuando terminan piden un nuevo GOP. El nodo coordinador les asigna el identificador del GOP siguiente o -1 si no hay más GOPs a codificar.

Una vez terminado el proceso de repartición y codificación el nodo distribuidor se encarga de integrar los GOPs codificados en un solo flujo.

Este proceso de asignación se puede observar en el grafo de estados de la figura 2.

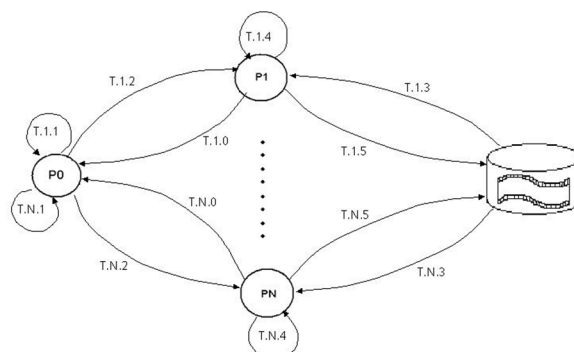


Figura 2. Distribución de carga (GOPs) por Demanda

Los estados del grafo de la figura 2 son descritos a continuación:

T.1.0. P1 avisa a P0 que está libre.	T.N.0. PN avisa a P0 que está libre.
--------------------------------------	--------------------------------------

T.1.1. P0 decide GOP a codificar
T.1.2. P0 avisa a P1 GOP a codificar
T.1.3. P1 lee GOP de disco
T.1.4. P1 codifica GOP
T.1.5. P1 escribe GOP codificado

T.N.1. P0 decide GOP a codificar por P1 por PN
T.N.2. P0 avisa a PN GOP a codificar
T.N.3. PN lee GOP de disco
T.N.4. PN codifica GOP
T.N.4. PN escribe GOP codificado

Parámetros del códec

Un problema a considerar con los códec de video es la gran cantidad de parámetros para afinar la codificación que depende del resultado que se espere. El mismo estándar establece en [7] varios perfiles de codificación que define valores específicos según el escenario donde se ocupará el video. En nuestro caso se usó el perfil "All Intra – Main" (AI-Main) que indica principalmente que todos los fotogramas se codificaran en modo Intra. El modo Intra se caracteriza por eliminar redundancia espacial (en un mismo fotograma) sin considerar si hay redundancia temporal. Lo anterior nos da facilidades para aplicar paralelismo de datos ya que permite separar fácilmente los GOPs. El variar el tamaño de los GOPs tiene como finalidad estudiar el efecto que puede producirse en la codificación aplicando los dos algoritmos de distribución de carga. En nuestro caso específico variamos el tamaño de los GOPs en 4, 8 y 12 fotogramas cada uno.

Otro parámetro importante es QP o cuantización. QP controla la cantidad de compresión para cada macrobloque en un fotograma. Un valor grande significa que habrá una cuantización más alta, es decir mayor compresión y por lo tanto menor calidad. Los valores más bajos significan lo contrario. En nuestro usamos valores de QP de 22 y 32.

Como ya se mencionó, para hacer dicha división de trabajo en la versión paralela sin variar los resultados que arroja la versión secuencial, se tomaron en cuenta las siguientes consideraciones:

- La unidad de división de trabajo es el GOP se codifican de acuerdo con un patrón establecido en el perfil de codificación.
- Los perfiles elegidos inicial con un fotograma I lo que hace GOPs independientes. Dichos perfiles oficiales fueron "encoder_lowdelay_main" que tiene un patrón IBBBIBBBIBBB y "IntraAll" que tiene solo fotogramas I.
- En base a lo anterior se determinaron 3 tamaños de GOPs (4, 8 y 12 fotogramas) como nuestro banco de pruebas.
- Por los tiempos tan altos de codificación se evaluaron solo con 2 videos de prueba en formato 4K (3840x2160) usando las secuencias estándar llamadas Marathon y Traffic and Buid usando solo 240 fotogramas.
- En cuanto al equipo donde se corrieron las pruebas, fueron llevadas a cabo en el servidor principal del clúster Agave que cuenta con 2 procesadores Xeon de 14 cores y 64 GB de RAM.
- Se lanzaron pruebas variando el número de núcleos de procesadores usando 5, 10, 15 y 20. Se corrieron 3 veces cada prueba para descartar pérdidas de eficiencia computacional debida a factores externos.

Resultados y discusión

En la figura 3 y 4 se muestran resultados con los videos Marathon y Traffic ambos con una resolución de 4K (3840x2160), usando en cada uno de ellos los algoritmos de preasignación (columna izquierda de ambas figuras) y demanda (columna derecha de ambas figuras) y con valores de cuantización (QP) de 22 (primera fila de cada figura) y 32 (segunda fila de cada figura). Se agregó una línea para mostrar el aprovechamiento ideal y ver qué tan lejos de ella se encuentra el aprovechamiento de las diferentes pruebas realizadas. En cada gráfica se puede observar las pruebas con diferente número de núcleos de procesador: 5, 10, 15 y 20. También en cada grafica se observa el aprovechamiento con 3 tamaños diferentes de GOPs (4, 8 y 12 fotogramas x GOP). En estas graficas podemos observar que el usar cuantización 22 o 32 no tiene un efecto significativo en comportamiento del aprovechamiento en ambos videos pues las gráficas son prácticamente las mismas con ambos valores de QP en cada uno de los videos, tanto si se usa preasignación como demanda. La diferencia básicamente es que la aceleración (speedup) fue ligeramente mayor con QP=32 y esto se debe a que la codificación aplicada devuelve un flujo comprimida de menor calidad.

Otra observación interesante es que en ambos videos el comportamiento de los algoritmos es muy similar por lo que se deduce que el contenido del video no fue significativo para el rendimiento de los algoritmos.

Por último se puede observar que el aprovechamiento del poder cómputo del algoritmo por demanda fue siempre mejor que el del preasignación. Sin embargo no se llegaron a los niveles de aprovechamiento que se obtuvieron versiones previas, con el códec H264 y con videos de menor resolución.

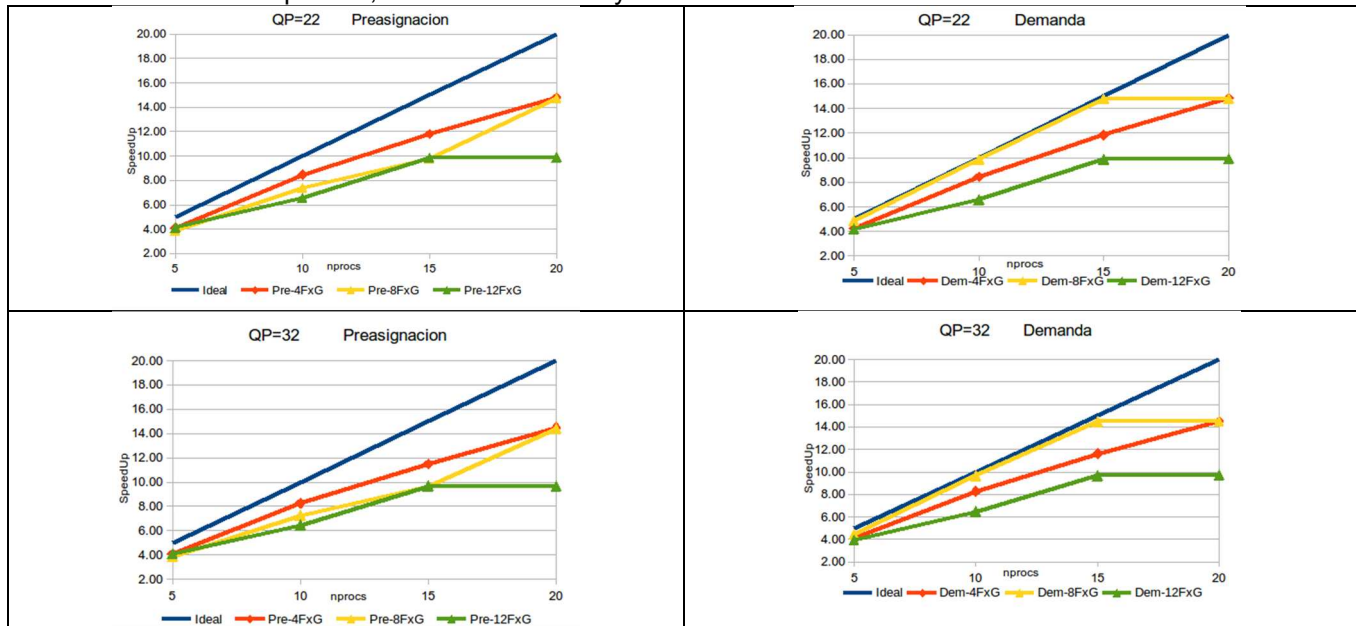


Figura 3. Aceleración (SpeedUp) del video Marathon en 4K (3840x2160) con cuantización en 22 y 32 y algoritmos de preasignación y demanda

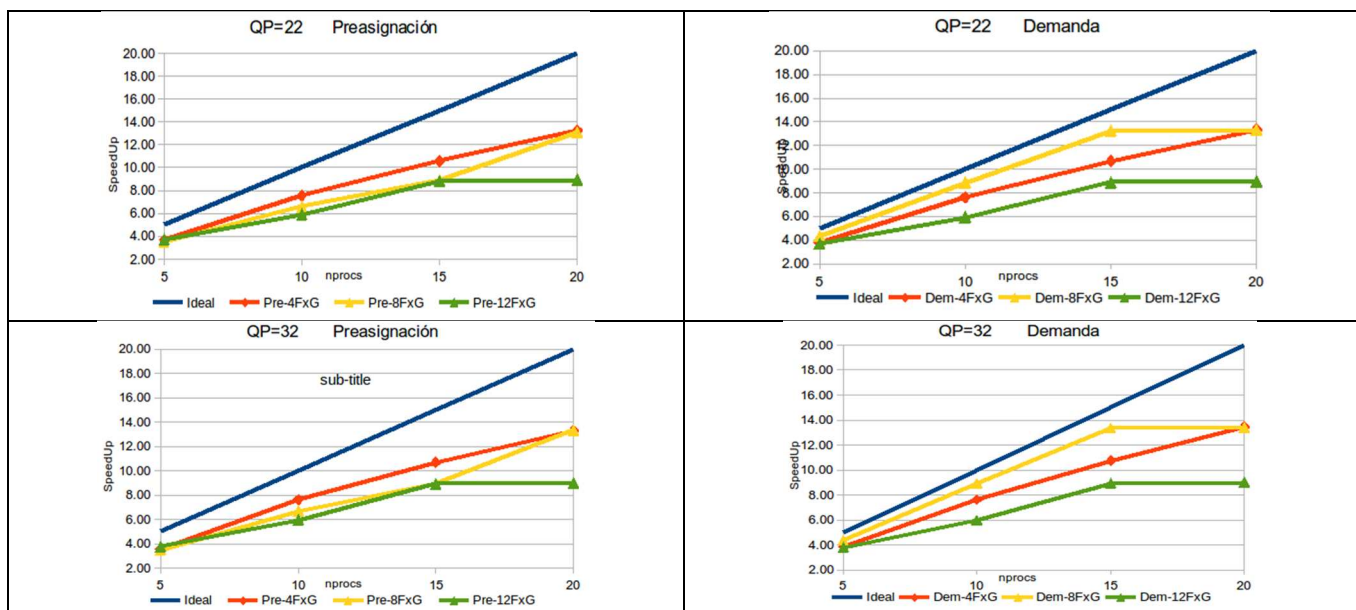


Figura 4. Aceleración (SpeedUp) del video Traffic and Builds en 4K (3840x2160) con cuantización en 22 y 32 y algoritmos de preasignación y demanda

Trabajo a futuro

Se desarrollará una versión del códec a partir de algoritmo por demanda donde se aplique paralelismo de grano medio usando hilos (threads) para realizar un particionado del fotograma a nivel de una baldosa (tile, es una región rectangular del fotograma) o rebanada (slice, región de macrobloques continuos no necesariamente rectangular).

Conclusiones

Es interesante observar como el codificar video de muy alta resolución (4K) con un códec muy demandante de recursos como el H265 regresa unos resultados muy diferentes de los que se habían obtenido en trabajos previos (de los mismos autores o de otros). Para comenzar el comportamiento del códec con un algoritmo que habido sido sumamente eficiente como el códec H264 ahora tiene un comportamiento más irregular. No son malos los resultados pero si es notorio que el H265 tiene partes que aprietan más la codificación en algunas secciones del video, lo que hace que algunos GOPs sean mucho más pesados que otros, a diferencia del H264 que tenía un comportamiento más uniforme. Fue relevante que el mejor comportamiento se obtuvo con tamaños de GOPs intermedio (8 fotograma s por GOP), pues se esperaba que fuera mejor con 12 fotograma s por GOP.

Por último, los resultados de este trabajo son además innovadores, pues no se han encontrado otros trabajos que evalúen versiones paralelas del códec H265 usando video 4K.

Agradecimientos

Los autores agradecen el apoyo del Tecnológico Nacional de México y a la Universidad Miguel Hernández (Alicante, España) para el desarrollo del presente trabajo de investigación

Referencias

- [1] K. Islas Lazcano Cristian y I. Sandoval Orozco, "Tecnología de la alta deficición en la televisión de alta definición", Tesis de Ingeniería mecánica y electrica, Instituto Politécnico Nacional unidad "Culhuacan", Culhuacan, Mexico, 2005.
- [2] Francisco Javier Montemayor-Ruiz , Miguel Ángel Ortiz-Sobrino . La producción de contenidos audiovisuales en ultraalta definición (UHD): Experiencia Inmersiva en el visionado multimedia en pantallas tv y smartphones. Journal of Communication, No. 12, 2016, pp. 41-57. Ed. Universidad de Salamanca. ISSN e: 2172-9077 , DOI: <http://dx.doi.org/10.14201/fjc2016124157>
- [3] Alexis Michael Tourapis, Gary Sullivan, Karsten Sühring, Tobias Oelbaum, Athanasios Leontaris, "H.264/14496-10 AVC Refence Software Manual", Documentación del Joint Video Team, Londres, Junio 2009.
- [4] Gary J. Sullivan, Jill M. Boyce, Ying Chen, Jens-Rainer Ohm, C. Andrew Segall, Anthony Vetro. Standardized Extensions of High Efficiency Video Coding (HEVC). IEEE Journal of Selected Topics in Signal Processing. Vol. 7. No. 6. PP: 1001 – 1016. DOI: 10.1109/JSTSP.2013.2283657 . Diciembre 2013,
- [5] Dan Grois, Detlev Marpe, Amit Mulayoff. Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders. Picture Coding Symposium 2013. San Jose California. Dec 8-11, 2013
- [6] Gary J. Sullivan ; Jens-Rainer Ohm ; Woo-Jin Han ; Thomas Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology Vol: 22 , Num: 12 . Dic. 2012 . PP 1649 – 1668. Print ISSN: 1051-8215
- [7] Bossen, F. Common HM test conditions and software reference configurations (JCTVC-L1100). Joint Collaborative Team on Video Coding. 2013
- [8] Rodríguez A, Pérez M., González A., Peinado J. y Fernández J.C.: "Paralelización del codificador H.264 con estimación de movimiento adaptativa en clústers de Pcs", XVI Jornadas de paralelismo, CEDI2005, Salamanca
- [9] Carlos Genis-Triana, Abelardo Rodríguez-Leon, Rafael Rivera-Lopez, Algoritmo Paralelo de Codificación de Video basado en H264/AVC con balanceo predictivo de carga, Programación Matemática y Software. Ed. UAEM, Cuernavaca, Mexico, Solcitud de ISSN : 00298, Vol. 2 Num. 2, pp.12-27, 2010
- [10] Rodríguez A.: "Desarrollo y evaluación de algoritmos paralelos para la compresión de secuencias de video", Tesis de Doctorado, Universidad Politécnica de Valencia, Noviembre del 2007
- [11] Carlos-Julian Genis-Triana, Abelardo Rodriguez-Leon, Rafael Rivera-Lopez, A comparison of two parallel algorithms using predictive load balancing for video compression, Recent Researches in Software Engineering, Parallel and Distributed Systems- SEPADS 2011, ISBN: 978-960-474-277-6, pp-200-205, Febrero 22, Cambridge, UK 2011
- [12] H. Migallón, V. Galiano, P. Piñol, O. López-Granado, M.P. Malumbres. Distributed memory parallel approaches for HEVCencoder. The Journal of Supercomputing. January 2017, Volume 73, Issue 1, pp 164–175. ISSN: 1573-0484.