Manejo Concurrente del Códec H.264/AVC para la Codificación de Video de Alta Definición en Tiempo Real

Gabriela Pérez-Reyes[1], Abelardo Rodríguez-León[2] y Rafael Rivera-López[3]

Departamento de Sistemas y Computación. Instituto Tecnológico de Veracruz
Calzada Miguel Ángel de Quevedo 2779, Veracruz, México
[1]savageheart50@hotmail.com [2]arleon@itver.edu.mx
[3]rrivera@itver.edu.mx

Resumen. En este artículo se muestran los resultados obtenidos de la paralelización por datos del codificador H.264/AVC versión 16.0 para video de alta definición. Con esta paralelización se logra codificación en tiempo real redimensionando una resolución de entrada de 640x400 pixeles en tres resoluciones de salida (640x400, 720x480 y 1280x720 pixeles), disminuyendo así el espacio de almacenamiento con respecto al uso de una resolución de entrada por cada resolución de salida; además durante las pruebas realizadas se comprobó que al utilizar el redimensionamiento se acelera ligeramente la codificación disminuyendo así la cantidad de procesadores necesarios para la codificación en tiempo real.

1 Introducción

La transmisión de flujos de video representa en la actualidad una de las aplicaciones más importante del internet en el mundo. Los usuarios de internet dedican muchas horas para observar video en tiempo real sobre deportes, música o noticias, así como la reproducción de videos por demanda almacenados en la red [1]. La estructura de un sistema típico dedicado a transmisión de flujos de video en tiempo real se presenta en la figura 1, donde se observa que una parte fundamental del proceso es la codificación del video para su transmisión por internet.

Por lo general los codificadores de video (codecs) comprimen la información para que pueda ser almacenada o transmitida ocupando el mínimo espacio posible. Para conseguirlo se aprovecha que las secuencias de video tienen redundancia espacial y temporal. En años recientes, varios estándares para codificación de video han sido desarrollados (H.264 [2], MPEG4 [3], JPEG2000 [4]) donde su principal objetivo es reducir el tamaño del video transmitido sin perder su calidad. La codificación de video requiere de sistemas de alto desempeño para alcanzar sus objetivos, por lo que el uso de sistemas multiprocesadores (clusters de computadoras, por ejemplo) para codificación de video se ha extendido en la actualidad [4].

M.A. Cruz-Chávez (Ed): CICos 2011, ISBN. 978-607-00-5091-6. pp. 192 – 203, 2011

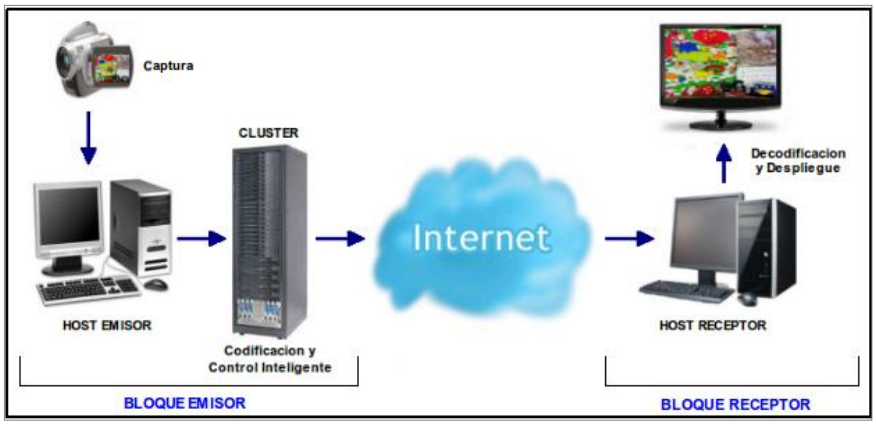

Fig. 1. Estructura de un sistema típico de transmisión de video en tiempo real.

Los codificadores de video han evolucionado mejorando sus características, calidad y tolerancia a fallos en medios de transmisión propensos a errores, sin embargo los requerimientos de poder de cómputo para comprimir el video han aumentado considerablemente [6]. Por ello es necesario disponer de mayor poder de cómputo como el que ofrecen los clústers de computadoras, con los cuales se puede realizar codificación en tiempo real.

La información de un video se compone de una secuencia de fotogramas llamados frames cuyo tamaño determina la resolución del video. Cada frame está formado de pixeles, los cuales se agrupan en macrobloques de 16x16 pixeles. Un conjunto de macrobloques forman un slice, el conjunto de Slices forman un frame y un grupo de frames forman lo que se denomina GOP (group of pictures) [7].

Existen varios enfoques ([8], [9], [10] y [11]) para el uso del codificador H264 en la transmisión de video en tiempo real. En este artículo se presenta un enfoque de paralelización por datos del H.264/AVC versión 16.0 para la codificación de video en tiempo real en varias resoluciones de salida sobre un clúster de alto desempeño. El resto de este documento se estructura de la siguiente forma: En la siguiente sección se hace un análisis de la versión secuencial de H.264/AVC y de la forma en que se puede reducir el tiempo de codificación. Después se presenta la propuesta de paralelización por datos del codificador para finalmente presentar el análisis de resultados y las conclusiones del trabajo.

2. Reducción del tiempo de codificación secuencial

Para poder disminuir el tiempo de codificación en paralelo es necesario disminuir en lo posible el tiempo de codificación de la versión secuencial del algoritmo. H.264/AVC tiene un archivo de configuración para definir las características de su ejecución, como lo son el patrón de codificación a usar, las dimensiones tanto del video de entrada como el de salida, el bit-rate que deberá tener el archivo de salida (al reproducirse), el tipo de perfil que se usará en la codificación, etc. Para cada perfil de

codificación (main, baseline y extended) la configuración tiene valores predeterminados. Para fines de mejorar el tiempo se codificación secuencial se modificaron los valores predeterminados en algunos de los parámetros en el perfil baseline, como se muestra en la tabla 1.

Tabla 1. Parámetros modificados de la versión 16.0

Parámetro	Descripción
FramesToBeEncoded	Indica la cantidad de frames totales a codificar sin tomar en cuenta el tipo de frames que se almacenan en una secuencia de video.
SourceResize OutputWidth OutputHeight	*SourceResize* determina si la secuencia de video de entrada se redimensiona o no. En este trabajo se utiliza la redimensión por lo que los parámetros *OutputWidth* y *OutputHeight* se utilizan para indicar el nuevo tamaño de la secuencia de video (640x400, 720x480 y 1280x720).
TraceFile ReconFile StatsFile	Se utilizan para realizar una depuración, para nombrar al archivo reconstruido y para activar estadísticas, respectivamente. Fueron desactivados para este trabajo buscando reducir el tiempo de codificación.
Verbose	Controla el nivel de información (0-3) acerca de la ejecución del codificador. En el nivel 0 despliega el número de frame que se está codificando, En el nivel 1 indica el número de frame, el tipo de frame que se está codificando, el valor del PSNR y otros valores. Los niveles 2 y 3 ofrecen aún más información específica acerca de la codificación en curso. El valor utilizado en este trabajo es de 0.
ProfileIDC	El parámetro indica el tipo de perfil que se está usando (66 para baseline que es el que se ocupa en este trabajo).
IntraPeriod	Indica el máximo periodo de frames I en la secuencia de codificación. Si el valor es 0 solo el primer frame de la secuencia es de tipo I. Si el valor es 1 se indica que el periodo es de 1, por lo que la secuencia tendrá un patrón de codificación igual a IIIIIIIIIIIII.
IDRPeriod	Indica el periodo de frames IDR. Este tipo de frames mejoran la codificación pero resulta más lenta la codificación. En este trabajo se determinó utilizar el valor 0.
PsliceSearch	Todos los parámetros *PsliceSearch* en sus búsquedas menores a 16x16, fueron desactivados, esto acelera la codificación.

Para codificar una secuencia de video los GOPs son procesados para eliminar las similitudes entre frames de video contiguos o para eliminar información irrelevante o redundante en los frames. Esta información redundante es eliminada aplicando modelos predictivos, redundancia sensorial o redundancia estadística [12].

Los modelos predictivos utilizan dos métodos para codificar frames: *intra-frame*, cuando los pixeles en un frame son codificados utilizando solo la información del mismo frame, e *inter-frame*, cuando un frame es codificado utilizando información de sus frames contiguos.

Los frames codificados utilizando codificación intra-frame se denominan "frames I o *Intra",* los frames que son codificados utilizando solo su frame predecesor se deno-

minan “frames P o predictivos” y los frames que son codificados utilizando tanto su frame predecesor como su frame sucesor se les denomina “frames B o bidireccionales”. De esto se observa que la codificación de un frame I debería consumir menos tiempo que el de un frame P o B.

En H.264/AVC, un GOP tiene 15 frames que pueden ser de tipo I, B y P. El patrón de codificación predeterminado en H.264/AVC es: IBBPBBP..., que permite codificar la secuencia de video produciendo un archivo muy pequeño, comparado con el tamaño original, pero con un alto tiempo de codificación.

Por ejemplo, la secuencia de video “mobile.yuv” [13], que es un video de 12.5 segundos, con 300 frames y una resolución de 720x480 que ocupa 148.32 MB, codificado con H.264/AVC en una computadora con procesador a 2.33 GHz, produce un archivo de 3.24 MB en 10,845 segundos (3 horas), muy lejos del tiempo real. Por eso es necesario modificar el patrón de codificación para disminuir el tiempo de codificación sin comprometer la calidad del video.

Se realizaron dos pruebas para determinar el esquema de codificación más adecuado que se debería aplicar al utilizar el H264/AVC de forma concurrente en un clúster de computadoras. La primera prueba radica en determinar el patrón de codificación más adecuado para la secuencia de video “mobile.yuv”; en este caso se utiliza la misma resolución de entrada (archivo original) como de salida (archivo codificado). La segunda prueba utiliza la ventaja proporcionada por la versión 16.0 de H264/AVC para codificar un video con una determinada resolución de entrada y obtener una resolución diferente de salida (modificando el parámetro SourceResize).

La tabla 2 presenta los resultados de la primera prueba, esto es, codificar con H.264/AVC el video “mobile.yuv” en tres resoluciones (640x400, 720x480 y 1280x720) utilizando tres patrones de codificación diferentes en una computadora de 2.33GHz.

Para cualquiera de los patrones seleccionados, el tamaño del archivo codificado es mucho menor, pero el tiempo de codificación es alto provocando un bit rate muy pequeño. Para producir codificación en tiempo real se debería alcanzar una compresión de 24 frames por segundo, por lo que en todos los casos presentados en la tabla 2 se observa que no se puede alcanzar tiempo real con un solo procesador y se hace necesario el uso de un clúster de computadoras y la definición de un esquema de paralelización de las secuencias de video.

En el caso del patrón de codificación predeterminado (IBBPBB...) se tiene que se requerirían 800 procesadores de 2.33 GHz trabajando concurrentemente para que se codifique en tiempo real la secuencia de video tomada como ejemplo. Para el patrón IPIPIPIPI... se necesitarían 93 procesadores y para el patrón IIIIIIIIIII... se deberían utilizar 17 procesadores.

Hasta este punto, el patrón de codificación IIIIIIIIIII... resulta más conveniente, ya que exige un número menor de procesadores para aplicar un esquema de paralelización por datos.

Tabla 2. Pruebas de codificación de mobile.yuv con diferentes patrones.

mobile.yuv (12.5 seg)			Resultados de la codificación secuencial			
Patrón de trama	**Resolución**	**Archivo original (MB)**	**Archivo codificado (MB)**	**Tiempo de codificación (seg)**	**Bit rate (Kbps)**	**Frames codificados por segundo**
IBBPBBP...	640x400	109.86	2.72	8078.91	2.76	0.04
	720x480	148.32	3.24	10845.18	2.45	0.03
	1280x720	395.51	5.91	29523.15	1.64	0.01
IPIPIPIPI...	640x400	109.86	8.64	854.66	82.83	0.35
	720x480	148.32	10.32	1143.85	73.92	0.26
	1280x720	395.51	29.11	1185.57	201.15	0.25
IIIIIIIIII...	640x400	109.86	13.78	147.47	765.42	2.03
	720x480	148.32	16.28	205.28	649.5	1.46
	1280x720	395.51	28.22	1185.57	473.11	0.61

La tabla 3 presenta los resultados de aplicar la codificación con el patrón IIIIIIIIIII... utilizando una resolución de entrada de 640x400 pixeles con tres secuencias de video (“pedestrian_area.yuv”, “stockhoml.yuv” y “mobile.yuv”) produciendo tres resoluciones de salida diferentes.

Tabla 3. Pruebas de codificación secuencial con redimensión activada.

Video	**Tamaño inicial (MB)**	**Frames**	**Resolución de salida**	**Tamaño codificado (MB)**	**Tiempo de codificación (seg)**	**Frames**	**CPU teóricos**
mobile	109.86	300	640x400	13.78	75.3	2.98	7
			720x480	13.98	89.32	3.36	8
			1280x720	14.37	184.88	1.62	15
pedestrian_area	137.33	375	640x400	5.78	73.2	5.12	5
			720x580	5.96	90.61	4.14	6
			1280x720	6.48	210.59	1.78	14
stockholm	221.19	604	640x400	11.76	85.66	7.05	4
			720x580	11.93	102.98	5.87	5
			1280x720	12.47	222.33	2.72	9

Se puede deducir que la codificación con redimensión desde una resolución de 640x400 muestra una mejoría en el tiempo y un aumento en la tasa de frames que se codifican cada segundo, reduciendo así la cantidad de procesadores teóricos para obtener tiempo real. Otra ventaja de usar la redimensión es que el tamaño de los archivo de entrada se reduce considerablemente al ser una sola resolución de 640x400 y no una resolución de entrada por cada resolución de salida.

3. Manejo concurrente del codificador H264/AVC

Como ya se indicó previamente, la codificación de frames I no presenta dependencia de datos con otros frames, a diferencia de los frames P y B que en su proceso de codificación si tienen una dependencia de datos. Con el patrón de codificación seleccionado IIIIIIIIIII... es posible entonces dividir la secuencia de video en grupos de GOPs para ser codificados de manera concurrente en varios procesadores, sin perder calidad de video y poder alcanzar el tiempo real. Aprovechando que la versión 16.0 de H264/AVC puede configurar la resolución de salida entonces, para este trabajo se utilizan tres grupos de nodos en un clúster para codificar de manera concurrente el video de entrada y producir tres secuencias codificadas de salida con diferente resolución, como se muestra en la figura 2. En esta figura se considera que cada secuencia de video de entrada está formada por 12 frames, por lo que se utiliza un nodo para codificar un frame.

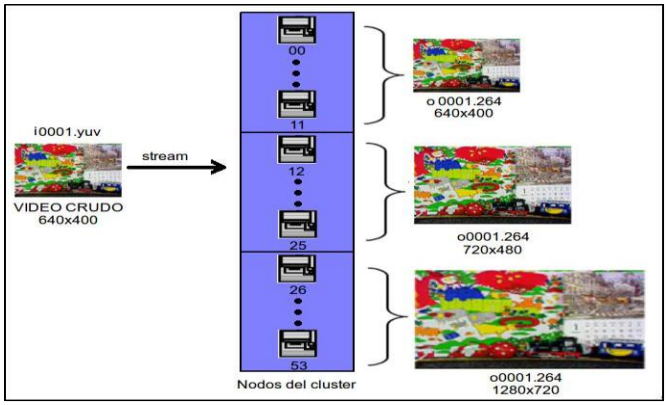

Fig. 2. Reparto de frames en un clúster de 54 nodos produciendo tres secuencias de video codificadas.

Debido a que la paralelización de GOPs por demanda demostró tener un mejor desempeño con respecto a las demás formas de distribución de GOPs evaluadas en [14], se decidió utilizar esta misma distribución de GOPs para la paralelización de datos de la versión 16.0. De esta forma un procesador está destinado únicamente a realizar el reparto de los GOPs mientras que el resto codifica un GOP a la vez. En la figura 3 se muestra un diagrama general del funcionamiento de la codificación paralela de GOPs por demanda utilizada en el presente trabajo.

De acuerdo con la figura 3 el proceso 0 se encarga únicamente de la distribución por demanda de los GOPs, los demás procesos reciben el número de GOP que les corresponde codificar (mensaje a en la figura 3) y toman del disco local el video correspondiente en formato .yuv (mensaje b), cada proceso codifican el GOP asignado generando un nuevo archivo con formato .264 (mensaje c). Los procesos que finalizan la codificación del GOP envían su rango al proceso 0 para que les asigne el siguiente numero de GOP a codificar (mensaje d) y así sucesivamente hasta que se detiene la codificación.

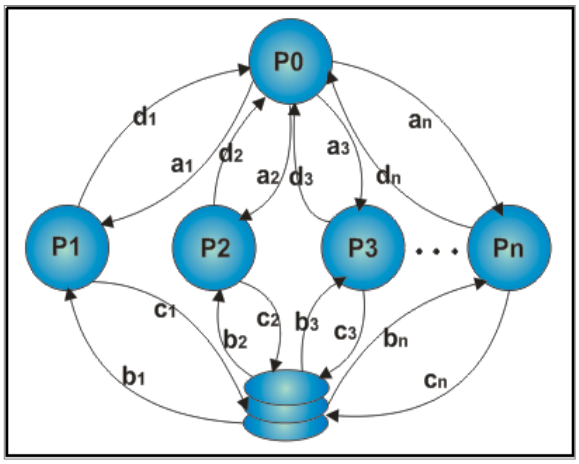

Figura 3.- Diagrama general del codificador paralelo por demanda con tres resoluciones.

4. Resultados obtenidos

Las pruebas del enfoque de paralelización por datos con múltiples resoluciones de H264/AVC se realizaron en el clúster Agave se encuentra en el Laboratorio de Cómputo Intensivo del Instituto Tecnológico de Veracruz (figura 4), que está formado por 28 núcleos de 2.33GHz con una arquitectura de 64 bits, 60 GB de memoria Ram y 1.2 TB de almacenamiento total. Se realizaron pruebas de rendimiento con 1,800 secuencias de video de 0.5 segundos (15 minutos de reproducción total). Los resultados mostrados en las figuras son el promedio de cinco corridas.

La figura 5 muestra la relación entre el tiempo de codificación y el número de procesadores que se obtiene con la resolución de 640x400 píxeles. Para codificar los vídeos en paralelo con esta resolución se han utilizado 8 procesadores, que es la mínima cantidad de procesadores necesarios para alcanzar tiempo real. En la figura 5 se aprecia que con todos los grupos de procesadores ocupados se obtiene tiempo real. El tiempo de codificación llega a reducirse a más de la mitad al alcanzar los 17 procesadores.

Para las resoluciones de 720x480 y 1280x720, las pruebas se inician con 9 procesadores y se van incrementando en intervalos de 4 procesadores. Se debe recordar que en la versión paralela del H.264/AVC presentada en este trabajo, un procesador se encarga de administrar GOPs entre los otros procesadores.

En la figura 6 se muestran las pruebas de rendimiento realizadas con la resolución de 720x480 píxeles donde se puede apreciar que a partir de 9 procesadores se consigue la codificación en tiempo real. Para la resolución de 720x480 píxeles a partir de 21 procesadores los resultados mejoran muy poco por lo que resulta un desperdicio de recursos usar más procesadores.

Figura 4.- Clúster Agave (izquierda) en el Laboratorio de Cómputo Intensivo del Instituto Tecnológico de Veracruz.

La figura 7 corresponde a las pruebas de rendimiento realizadas a los vídeos con resolución de 1280x720 píxeles donde se puede observar que con 9 procesadores no se obtiene tiempo real y es necesario aumentar el número de procesadores a 21 para alcanzar el tiempo deseado.

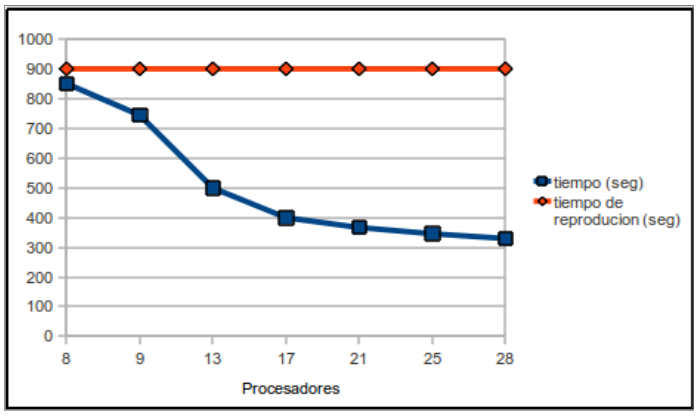

Figura 5.- Rendimiento para secuencias de video con una resolución de 640x400.

Las pruebas que se han descrito en las figuras 5, 6 y 7 fueron realizadas con la versión del codificador paralelo produciendo una sola resolución de salida.

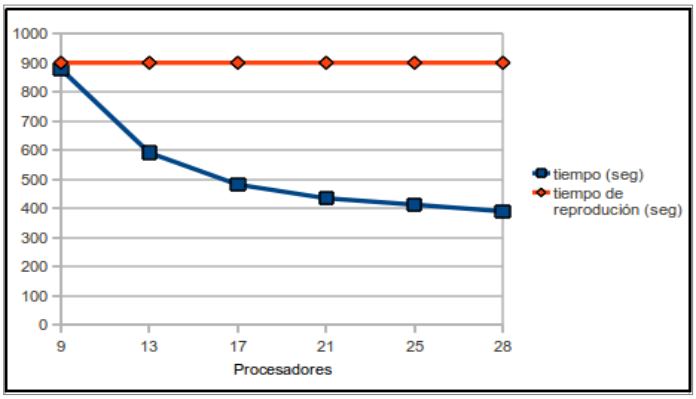

Figura 6.- Rendimiento para secuencias de video con una resolución de 720x480.

En la figura 8 se muestran los resultados obtenidos en la pruebas de rendimiento con resoluciones de salida de 640x400 - 720x480 píxeles aplicando la versión multi-resolución del codificador de video. El tiempo real se alcanza usando la combinación de 18 procesadores: 1 que distribuye, 7 para codificar la resolución de 640x400 y 10 para codificar la resolución de 720x400.

En todos los casos anteriores se usa paralelismo de datos y aunque en teoría el programa de codificación debería ser tan eficiente como procesadores disponibles se tengan, en la práctica esto no resulta de esta manera, debido a que al usar más procesadores el tiempo de codificación de las secuencias de video individuales aumenta. Se considera que el rendimiento se ve comprometido debido al acceso constante al disco duro.

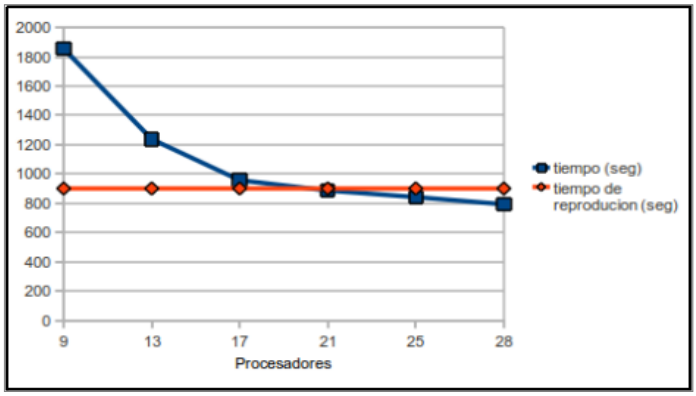

Figura 7.- Rendimiento para secuencias de video con una resolución de 1280x720.

El servidor del clúster Agave comparte el espacio en disco duro mediante NFS por lo que se replican los datos entre todas las computadoras que se encuentran ejecutando el programa, por eso, entre más computadoras estén escribiendo datos en el disco duro, mayor es la carga que tiene el servidor NFS y provoca que el tráfico vaya más len-

to. Se piensa que esta situación podría mejorar si se cambiara el sistema de compartición de archivos NFS al uso de XFS o incluso utilizando un array de discos duros. Otra forma se resolver este problema podría ser modificando la arquitectura de la red interna a una más eficiente, actualmente es ethernet, pero podría cambiarse a una arquitectura como la de InfiniBand, Myrinet, etc.

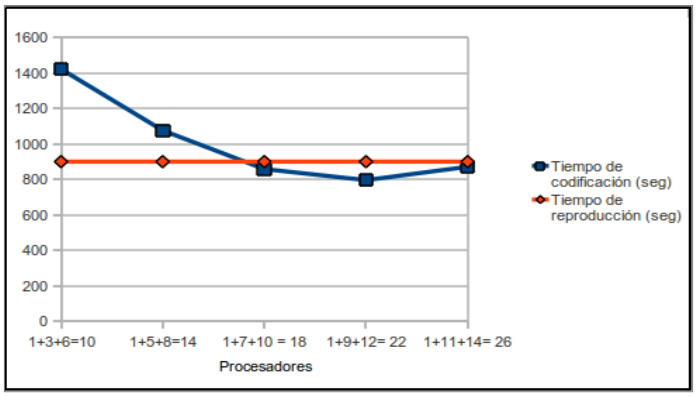

Figura 8.- Rendimiento obtenido usando la versión multiresolución con resoluciones de 640x400 y 720x480.

Con la combinación de las resoluciones de 640x400-1280x720 y 720x480-1280x720 no se obtuvo tiempo real al aplicar el esquema de paralelización de video por demanda multiresolución.

En la figura 9 se muestra el speed up obtenido en las pruebas al usar diferentes combinaciones de procesadores para codificar con resoluciones de salida de 640x400 - 720x480 píxeles aplicando la versión multiresolución del codificador de video. El speed up es una medida para medir la eficiencia de los programas paralelos que se obtiene al dividir el tiempo de ejecución de un programa secuencial entre el tiempo de ejecución del mismo programa en paralelo.

Con las primeras combinaciones, el speed up se va incrementando aparentemente de forma gradual obteniendo el máximo speed up con 22 procesadores, pero al aumentar más procesadores éste decae, esto puede estar relacionado con el sistema para compartición de archivos que maneja el clúster o por la arquitectura de la red interna.

Conclusiones y trabajos futuros

Como se observa en los resultados presentados en la sección anterior, es posible obtener codificación en tiempo real para varias resoluciones de secuencias de video en un clúster de computadoras implementando un esquema de distribución de carga por demanda. Al implementar la redimensión de los videos de salida desde una resolución 640x400 píxeles se reduce el espacio en disco duro necesario para el almacenamiento de los videos de entrada.

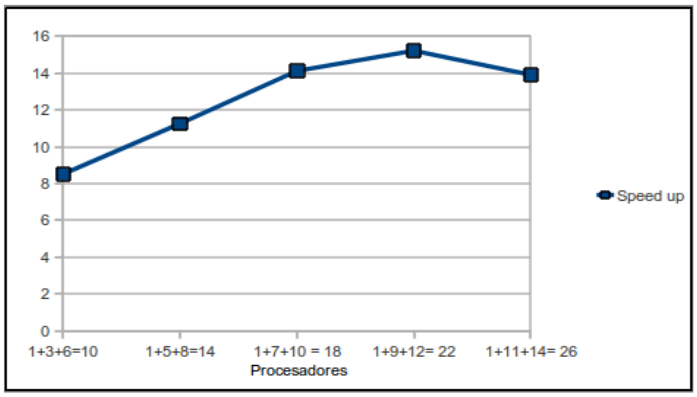

Figura 9.- Speed up de la versión multiresolución con las resoluciones de 640x400 y 720x480.

Se logra codificación en tiempo real redimensionando la secuencia de video desde 640x400 a 1280x720 pero con la salvedad de que esta codificación no puede realizarse de manera simultánea con las demás resoluciones debido a la cantidad de procesadores disponibles en el clúster Ágave. Con el codificador H.264/AVC v. 16.0 paralelo incrustado, en teoría se necesitarían 36 procesadores en el clúster Agave para lograr codificación en tiempo real redimensionando desde la resolución de entrada de 640x400 píxeles a las resoluciones de salida de 720x480 y 1280x720 píxeles simultáneamente junto con la codificación de 640x400 sin redimensionamiento.

Para trabajos futuros se propone realizar una paralelización más profunda, utilizando grupos de GOPs y slices para hacer la codificación aún más eficiente.

Finalmente debe aclararse que los valores mostrados en las pruebas finales no son valores absolutos, pertenecen únicamente a la codificación realizada con el clúster Agave. Por ello en versiones posteriores se podría implementar un proceso que determinara la cantidad de procesadores necesarios para la codificación en tiempo real con un hardware determinado.

Referencias

[1]. D. Wu, Y.T. Hou, W. Zhu, Y-Q. Zhang y J. M. Peha: *Streaming Video over the Internet: Approaches and Directions*, en IEEE Trans. on Circuits and Systems for Video Technology, vol. 11, no. 3, 2001, pp. 282-300.

[2]. T. Wiegand, G.J. Sullivan, G. Bjontegaard y A. Luthra: *Overview of the H.264 / AVC Video Coding Standard*, en IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, 2003, pp. 560-576.

[3]. T. Ebrahimi y C. Horne: *MPEG-4 natural video coding - An overview*, en Signal Processing: Image Communication, vol. 15, 2000, pp-365-385.

[4]. M. J. Gormish, D. Lee y M.W. Marcellin: *JPEG 2000: Overview, architecture and applications*, en Proc. IEEE Int. Conf. of Image Processing, Vancouver, Canada, 2000, vol. II, pp. 29-32.

[5]. J.C. Fernandez y M.P. Malumbres: *A parallel implementation of H.26L video encoder*, en Lectures Notes on Computer Sciencies, vol. 2400, 2002, pp. 830-833.

[6]. E.G. Richardson: *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*, Editorial Wiley, 2003.

[7]. A. Tourapis, G. Sullivan, K. Sühring, T. Oelbaum y A. Leontaris: *H.264/14496-10 AVC Refence Software Manual*, Documentación del Joint Video Team, Londres, R.U., 2009.

[8]. D.J. Wu, M.J. Chan, J.P. Chen y J.Y. Tham: *A real-time H.264 video streaming system on DSP/PC platform*, en IEEE 13th Int. Sym. On Consumer Electronics, Singapore, Singapore, 2009, pp.527-530.

[9]. A. Foncubierta-Rodríguez y J. R. Cerquides-Bueno: *Análisis de calidad de vídeo H.264 en streaming sobre HSUPA*, en Memorias del XII Congreso Iberoamericano de Internet, Telecomunicaciones y Sociedad de la Información, Madrid, España, 2009, pp. 401-407.

[10]. A. Rodríguez, M. Pérez, A. González, J. Peinado y J.C. Fernández J.C.: *Paralelización del codificador H.264 con estimación de movimiento adaptativa en clusters de PCS*, en Memorias de las XVI Jornadas de Paralelismo, Salamanca, España, 2005.

[11]. A. Rodríguez, A. González y M.P. Malumbres: *Hybrid Parallelization of an d H.264/AVC Video Encoder*, en Memorias de las XVII Jornadas de Paralelismo, Albacete, España, 2006.

[12]. E. G. Richardson: *Video Coding Design*, Wiley, 2002.

[13]. Xiph.Org Foundation: *Video test sequences*, en http://media.xiph.org/video/derf/, 2011.

[14]. A. Rodríguez: *Desarrollo y evaluación de algoritmos paralelos para la compresión de secuencias de video*, Tesis de Doctorado, Universidad Politécnica de Valencia, España, Noviembre del 2007.