Evaluación de una versión paralela para el Codec H.264/AVC

Carlos Genis Triana, Abelardo Rodríguez León

Departamento de Sistemas y Computación, Instituto Tecnológico de Veracruz,
Calzada Miguel Ángel de Quevedo 2779, Veracruz, México
carlosgenis@yahoo.com.mx, arleon@itver.edu.mx

Resumen. En este capítulo se presenta una evaluación del codificador de video basado en el estándar H.264/AVC propuesta por JVT contra una versión paralela implementada por GAP de DISCA-UPV. El primero realiza la tarea de manera secuencial mientras tanto el segundo segmenta y distribuye paralelamente una secuencia de video (GOPs) entre los nodos de un cluster. Para la distribución de la carga utilizaron el estándar MPI, desarrollada por el grupo Forum de MPI. Se observó un patrón de codificación no equivalente entre ambas versiones y una codificación no cíclica en la versión paralela, con lo cual el número de frames es una limitante para realizar pruebas de codificación con mayor número de GOPs. Las pruebas realizadas y analizadas en este capítulo, son el punto de partida que determinará el rumbo que seguirá la presente investigación a corto plazo. En dicha investigación se buscará implementar y evaluar un algoritmo paralelo con balanceo de carga para la compresión de secuencias de video, usando el codificador H.264, buscando tiempo real en la codificación, así como determinar el tipo de video más adecuado para esta versión.

1 Introducción

Existen muchas aplicaciones para la transmisión de imagen digital en movimiento tales como la videoconferencia, el control y monitorización de sistemas robotizados, la telemedicina, los canales de difusión en Internet o los sistemas de video por demanda. Sin embargo, la transmisión de imagen digital en movimiento se encuentra con el problema de que requiere el almacenamiento y procesamiento de datos multimedia. Debido a la naturaleza de estos datos, se necesita un ancho de banda excesivo y que hoy en día no está disponible para la mayoría de los usuarios. Por ejemplo, para transmitir video crudo (sin codificar) en una resolución de 176x144, utilizando 24 bits de profundidad y una tasa de 10 frames/seg, se necesitaría un ancho de banda de casi 6 Mbps. Además del problema del ancho de banda también existe el inconveniente del espacio utilizado para almacenar dicho video. Así, un segundo de un video de color con el estándar NTSC requiere casi 23 MBytes y un video de 90 minutos, en este mismo formato, requeriría aproximadamente 120 GBytes. Por lo tanto, se tiene que reducir la cantidad de información que generan las imágenes en movimiento a través del uso de técnicas de compresión de datos respetando, en la medida de lo posible, su calidad.

A. Gelbukh, G. Sidorov, W. Olán, J. Vera (Eds.) Recientes avances en la ciencia de la computación México, 155-166.© Centro de Investigación en Computación, IPN, México

Para comprimir las imágenes digitales lo que se hace es eliminar la información redundante. Para ello, los sistemas de compresión se aprovechan de los tres tipos de redundancia existentes en los videos digitales:

- *Espacial*. Normalmente, los valores entre píxeles vecinos en una imagen son similares.
- *Espectral.* En imágenes compuestas por más de una banda espectral, los valores espectrales de un mismo píxel están fuertemente correlacionados.
- *Temporal*. Frames consecutivos en una secuencia de video presentan frecuentemente pocos cambios.

La redundancia espacial y la redundancia espectral se tratan a nivel de imágenes, ya que explotan la redundancia implícita en las imágenes, mientras que la redundancia temporal es explotada por técnicas (como predicción y compensación de movimiento) que sólo codifican las diferencias entre frames adyacentes pertenecientes a una secuencia de video.

En los últimos años se han propuesto y se han desarrollado una variedad de estándares de compresión de video e imagen (H.26X, MPEG-X, JPEG200,...). Sin embargo, no todos los estándares proporcionan las mismas características. Unos codificadores obtienen mayor calidad de imagen, a costa, sobre todo, de un mayor tiempo de procesamiento, por lo que se ha tenido que buscar un equilibrio entre el tiempo de procesamiento (tiempo de codificación) y la calidad que se obtiene de las imágenes decodificadas. Por tanto, si se aumentara la velocidad de procesamiento se podrían utilizar codificadores que ofrecen una calidad mayor de las imágenes y de las secuencias de video. Para conseguir una capacidad de procesamiento muy alta se tienen varias alternativas:

- *Sistemas fuertemente acoplados.* Como el proceso de codificación suele requerir mucho poder de cómputo generalmente se logra, usando computadoras de altas prestaciones que ayuden con sus sistemas multiprocesadores integrados en un solo gabinete, llevar a cabo este proceso con tiempos de retardo aceptables. El inconveniente es que son equipos muy costosos.
- *Tarjeta codificadora.* Desafortunadamente son pocas las tarjetas codificadoras que pueden dar un buen rendimiento en tiempo real (codificación menor o igual a la duración de la secuencia de video). Las que lo logran lo hacen con estándares como MPEG1 o MPEG2 como la Broadway de DataTraslation. Existen tarjetas que codifican también en MPEG4 aunque no lo hacen en tiempo real y con configuración limitada.
- *Sistemas débilmente acoplados.* Una alternativa que no requerirá usar costosos equipos es usar cluster de computadoras personales conectadas por una red, las cuales se han hecho muy populares ya que ofrecen muy buenas prestaciones a un precio más económico y con mayor escalabilidad que los sistemas fuertemente acoplados. Para lograr esto hay que implementar los algoritmos de compresión usando técnicas de programación paralela.

En este trabajo de investigación se presenta una evaluación del codificador de video basado en el estándar H.264/AVC propuesta por el Joint Video Team (JVT)

contra una versión paralela implementada por el Grupo de Arquitecturas Paralelas (GAP) del Departamento de Informática, Sistemas de Cómputo y Automática de la Universidad Politécnica de Valencia (DISCA-UPV) [1]. Cabe señalar que la paralelización de la carga la hicieron utilizando la técnica estándar de programación paralela para el paso de mensajes (MPI), desarrollada por el grupo Forum de MPI

En el punto 1 se hace una introducción a la codificación de video. En los puntos 2 y 3 se hacen descripciones generales de los estándares MPI y H.264/AVC, respectivamente. En el punto 4 se enuncian las características de la versión paralela. En el punto 5 se muestran algunos resultados obtenidos de la evaluación del codificador secuencial contra el paralelo y finalmente en el punto 6 se hace mención sobre el trabajo futuro que se tiene contemplado realizar tomando como punto de referencia las pruebas citadas en este documento.

2 MPI

MPI fue desarrollado por el Forum MPI, que es un grupo abierto en representación de una amplia sección de la industria y de intereses académicos. MPI es un conjunto de funciones (API) que permite a los programadores escribir aplicaciones paralelas. Estas aplicaciones están formadas por una serie de procesos que intercambian mensajes (operaciones cooperativas) a través de MPI para llevar a cabo un trabajo común. Esta librería de paso de mensajes está especificada para C, C++ y Fortran.

MPI se puede utilizar tanto en sistemas fuertemente o débilmente acoplados. Sin embargo, hoy en día, se suele utilizar fundamentalmente con cluster de computadores, ya que permiten obtener una gran potencia de computación a un bajo costo. Debido a la gran variabilidad de entornos donde se utiliza MPI, se puede deducir que el estándar es portable e independiente de la plataforma en la que se ejecuta.

La especificación está dividida en dos partes: MPI-1 (versión 1.2) y MPI-2 (versión 2.0). En un principio fue diseñada para permitir el desarrollo de librerías paralelas, pero actualmente también proporciona un acceso paralelo al hardware tanto a usuarios finales, como a desarrolladores de librerías y de aplicaciones.

MPI utiliza el paso de mensajes porque es un paradigma de programación fácilmente entendible, eficiente y que emplean múltiples aplicaciones. Además es ideal para desarrollar programas paralelos que sean portables o para aumentar la productividad de un sistema secuencial. Además las características de MPI le hacen ser un sistema muy completo: modular (ideal para las librerías), portable, dispone de comunicaciones seguras, permite crear subgrupos y existen herramientas para medir el rendimiento que ofrece.

3 H.264

H.264 es uno de los estándares de codificación de video más nuevos desarrollado por el VCEG (*Video Coding Experts Group*) y por el MPEG (*Moving Picture Experts Group*). El VCEG es un grupo que pertenece a la ITU-T (*International Communications Union*), mientras que el MPEG pertenece a la ISO/IEC

(*International Standarization Organization/International Electrotechnical Commission*).

El objetivo principal por el cual surgió el estándar H.264 fue el de aumentar la tasa de compresión de video. Antes de salir este estándar, existían otros codificadores de video como el MPEG-2. Este codificador se utilizaba básicamente para la transmisión de señales de alta definición (HD) y de definición estándar (SD) sobre satélite o cable y para el almacenamiento de señales de SD en DVDs. Pero, debido al crecimiento de la popularidad de la televisión de alta definición, se hace necesario utilizar técnicas de codificación más eficientes que MPEG-2. Además, existen otros medios de transmisión (xDSL, UMTS) que ofrecen tasas de transferencia mucho menores que los canales broadcast y, por lo tanto, necesitan de una mayor tasa de compresión.

En 1998 el VCEG publicó un proyecto (H.26L) que tenía el propósito de doblar la eficiencia de la codificación que obtenían los estándares de codificación que había en ese momento. En octubre de 1999 salió la primera versión de este proyecto. En diciembre del 2001, el VCEG y el MPEG se unieron y formaron el JVT (Joint Video Team) con el objetivo de desarrollar un estándar de codificación de video nuevo. Fruto de esta unión, en marzo del 2003, se publicó el H.264 /AVC.

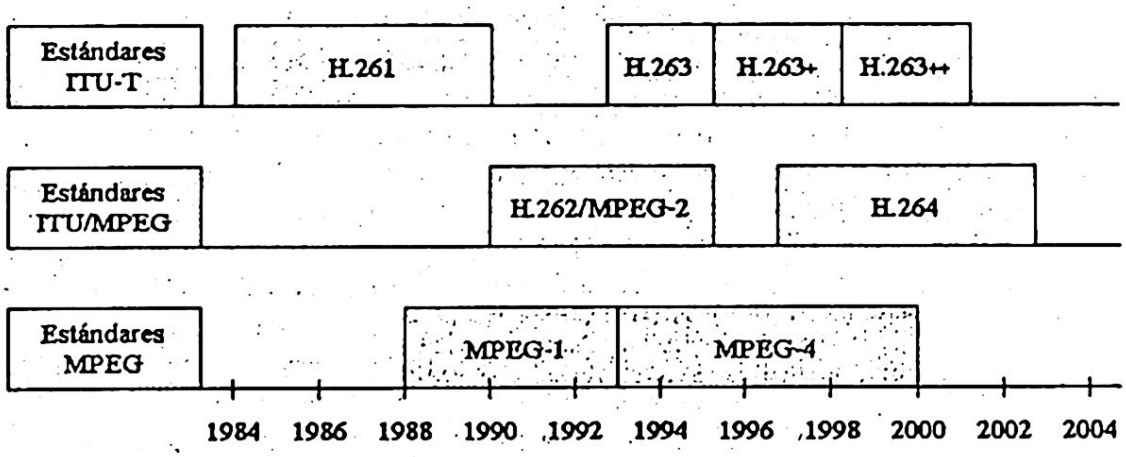

Fig. 1. Evolución de las recomendaciones ITU-T y estándares MPEG

En la figura 1 se muestra la evolución de las recomendaciones propuestas por la ITU-T y de los estándares de la ISO/IEC. Las recomendaciones de la ITU-T (denotadas como H.26x) fueron diseñadas principalmente para aplicaciones de video en tiempo real. Mientras que los estándares de la ISO/IEC (denotados como MPEG-x) están más enfocados a cubrir las necesidades de almacenamiento de video (DVD), multidifusión de video (Cable, DSL, TV por satélite) y streaming de video (video en Internet o sobre redes inalámbricas). La mayor parte de los estándares y recomendaciones han sido desarrollados por estos dos comités de forma independiente, exceptuando los estándares H.262/MPEG-2 y H264 que fueron desarrollados conjuntamente.

Algunas de las ventajas que presenta el estándar H.264 frente a los estándares anteriores son:

- *Ahorro de tasa de bits superior al 50%.* Comparado con el H.263+ o con el MPEG-4, el estándar H.264 permite reducir la tasa de bits más de un 50%.

- *Vídeo de alta calidad.* Ofrece video de buena calidad tanto con tasas de bits altas como bajas.
- *Tolerancia a fallos.* El H.264 ofrece herramientas para trabajar correctamente cuando se pierden paquetes en la red o se producen errores a nivel de bits en redes inalámbricas.
- *Entorno de red amigable.* Debido a la capa NAL, se pueden transportar flujos de bits de forma sencilla.

4 Versión paralela del H.264

En este apartado se hace una descripción de una implementación paralela desarrollada por el GAP. El objetivo de esta versión fue conseguir un codificador de video con mayor rendimiento que el propuesto por el JVT, mediante el reparto de la carga de trabajo entre varios procesadores. Estos últimos cooperarán en la codificación consiguiendo así una significativa reducción del tiempo de ejecución que permitirá la codificación de video en tiempo real. Eligieron este estándar porque hasta el momento es la última versión propuesta por el JVT; además proporciona video codificado de alta calidad y el costo computacional de este codificador es bastante alto.

Como ya se sabe, los flujos de video están formados por una secuencia de cuadros o *frames* linealmente distribuidos en el tiempo. La cantidad de trabajo que conlleva la codificación de cada cuadro es variable y en la mayor parte de los casos impredecible.

Dada la naturaleza del procesamiento paralelo, y más concretamente el procesamiento paralelo mediante paso de mensajes, los mayores beneficios se obtienen balanceando lo máximo posible la carga de cómputo entre los distintos procesadores. Intentando conseguir un reparto lo más equitativo posible. Y, al mismo tiempo, intentar reducir las comunicaciones necesarias entre ellos. Esta característica, aunque deseable, no es siempre posible. Como límite adicional al aumento de las prestaciones, la parte paralelizable de un algoritmo casi nunca supone el 100% de éste, lo que limita el *speed-up*, o reducción del tiempo de ejecución posible. Adicionalmente, las comunicaciones entre procesadores conllevan siempre una sobrecarga muchas veces considerable.

4.1 Segmentación de la carga basada en GOPs

El reparto de la carga es la distribución de la secuencia de *frames* entre los distintos procesadores. Para que cada procesador pueda llevar a cabo su tarea de la forma más independiente posible, no debería existir interdependencia entre la codificación de una imagen y otra, o bien, debería de ser mínima. Evidentemente, esta premisa no se cumple, ya que el estándar de codificación de video trata de aprovechar la redundancia temporal existente en la secuencia. Con ello consigue mayor eficiencia en la codificación. Por esta razón, en una primera implementación paralela, definieron como la unidad mínima de carga a repartir entre los procesadores la secuencia de imágenes que constituyen un GOP (*Group of pictures*) en lugar de un único *frame*. Cada uno de los GOPS a codificar se puede tratar de forma independiente, es decir, no

existe ninguna interdependencia entre un GOP y otro a la hora de su codificación. Esto se debe principalmente a que cada GOP comienza con un cuadro de tipo I. Este tipo de *frame* se codifica de forma independiente. El resto de cuadros se codificará con información recogida a partir del cuadro I o bien de alguno derivado de éste. Por tanto el GOP es una unidad independiente, en cuanto a la codificación concierne.

Ya que en principio el costo computacional de la codificación de los cuadros que componen un GOP es variable e impredecible, no se puede hacer ninguna distinción entre un GOP y otro. Lo que dificulta la posibilidad de hacer un reparto equitativo del trabajo. Es por esto que simplificaron en este algoritmo, asumiendo que la tarea tiene un costo idéntico de cómputo para cada GOP. Bajo esta premisa utilizaron el siguiente esquema de reparto: la secuencia de video se divide en GOPs, cada uno de los cuales contendrá una secuencia de cuadros. A priori esta secuencia puede ser arbitraria con tal de que se cumpla la premisa de que el primer cuadro a codificar sea siempre un cuadro de tipo I. Por tanto la secuencia de video estará formada por un número determinado de GOPs.

Los procesadores serán etiquetados con un índice que se moverá en el conjunto de los enteros entre el 0 hasta el número de procesadores disponibles menos 1. Se divide aritméticamente el número de GOPs entre el número de procesadores, y se asigna a cada procesador el número de GOPs consecutivos resultante de manera secuencial. El número de GOPs restante (n) de la división entera entre los procesadores se asignarán a los n primeros procesadores, a razón de un GOP por procesador.

Esta división de la carga es de grano grueso, es decir, la unidad mínima de carga que corresponde a cada procesador es considerablemente grande. Este tipo de paralelización tiene ciertos inconvenientes y también algunas ventajas. La principal ventaja de este tipo de reparto es que minimiza la frecuencia con la que hay que realizar comunicaciones entre los distintos procesos. Y por tanto la sobrecarga que introducen las comunicaciones es menor.

En este caso, cada procesador realiza todo su trabajo asignado sin intercambiar información. Únicamente en una última fase de recomposición de las diferentes partes del video codificándose intercambiarán información. El gran inconveniente es que, si los datos de entrada se generan de forma secuencial en tiempo de codificación, es posible que los procesadores tengan que esperar un tiempo considerable a la recepción de los mismos. Esta técnica puede servir para acelerar la codificación de secuencias de video previamente almacenadas. Sin embargo, carece de utilidad para secuencias de video generadas en tiempo real, como en una videoconferencia.

5 Resultados obtenidos

La versión paralela anteriormente citada ha sido sometida a pruebas de evaluación a fin es establecer un punto de referencia para una futura implementación mejorada de dicho codec.

El *banco de pruebas* consiste en un cluster Aldebaran.upv.es, el SGI Altix 3000 de la UPV (Universidad Politécnica de Valencia). Este es un sistema de 48 procesadores Itanium II con memoria distribuida NUMA, con conexión directa a la SAN (red de almacenamiento) del CPD (centro de proceso de datos de la UPV) y sistema operativo

Linux RedHat. La *carga de entrada* consiste en 6 secuencias de video de dominio público en formato YUV 4.2.0. Sus características se enuncian en la tabla 1.

Tabla 1. Atributos de las secuencias de video empleadas para la evaluación

Secuencia (Sec)	Formato	Frames	Tamaño	Fondo	Fig. Central	Objs. Mov
1. foreman	qcif (176x144)	300	11,404,800	1	1	1
2. students	qcif (176x144)	1007	38,282,112	0	1	2
3. foreman	cif (352x288)	300	45,619,200	1	1	1
4. students	cif (352x288)	1007	153,128,448	0	1	2
5. martin	720x480	240	124,416,000	1	1	1
6. ayersroc	720x480	240	124,416,000	1	2	3

El *patrón de codificación* utilizado para todas las pruebas secuenciales se muestra en la tabla 2; por otro lado, el patrón de codificación utilizada para las pruebas paralelas se proporciona en la tabla 3.

Tabla 2. Patrón de codificación secuencial: P0 – Codifica 31 Frames

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I	B	B	P	B	B	P	B	B	P	B	B	P	B	B	I
0			**1**			**2**			**3**			**4**			**5**
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
B	B	P	B	B	P	B	B	P	B	B	P	B	B	I	
		6			**7**			**8**			**9**			**10**	

El patrón de codificación secuencial es exactamente la secuencia de compresión deseada; sin embargo, en la versión implementada por GAP se tiene esa limitante, ya que no es posible obtener el mismo patrón que el codec secuencial. El último Frame del GOP 0 deberían ser tipo I (Frame 15) y ser codificado como primer Frame del GOP 1. El resto de los Frames del GOP 1 debería seguir la misma secuencia que el GOP 0 de tal forma que se tenga 15 frames por cada GOP. Lo anterior se observa en las tablas 3 y 4.

Cabe mencionar que la obtención de dicho patrón es importante por dos razones: para tener resultados equivalentes (comparables) se requiere el mismo patrón de codificación tanto en el codec secuencial como en el paralelo y además este patrón es el idóneo para obtener la compresión de video manteniendo un equilibrio entre *tiempo y calidad.*

Tabla 3. Patrón de codificación paralelo

P0 – Codifica GOP 0 (16 Frames)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I	B	B	P	B	B	P	B	B	P	B	B	P	B	B	P
0			**1**			**2**			**3**			**4**			**5**

P1 – Codifica GOP 1 (16 Frames)

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
I	B	B	P	B	B	P	B	B	P	B	B	P	B	B	P
6			**7**			**8**			**9**			**10**			**11**

Tabla 4. Patrón de codificación paralelo deseado

P0 – Codifica GOP 0 (15 Frames)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I	B	B	P	B	B	P	B	B	P	B	B	P	B	B
0			**1**			**2**			**3**			**4**		

P1 – Codifica GOP 1 (15 Frames)

15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
I	B	B	P	B	B	P	B	B	P	B	B	P	B	B
6			**7**			**8**			**9**			**10**		

Para evaluar el aprovechamiento del algoritmo se ha obtenido el tiempo de ejecución secuencial (T_s) y paralelo (T_p). Los tiempos tomados comprenden: *carga inicial del proceso, lectura de disco, codificación, escritura de disco y las comunicaciones.*

Tabla 5. Tiempos de codificación secuencial y paralelo en segundos

Sec.	Ts	Tp-2P	Tp-4P	TP-6P	Tp-8P	Tp-10P	Tp-12P	Tp-14P
1	306.12	147.45	84.37	63.25	42.27	42.35	42.03	21.08
2	306.94	148.05	83.94	63.01	41.95	41.99	41.80	21.05
3	1258.00	598.60	341.70	257.33	171.59	171.39	172.03	85.78
4	1218.12	591.63	336.17	254.74	168.84	169.43	168.22	85.06
5	4216.10	2060.01	1196.77	895.68	605.17	600.16	595.81	300.15
6	4268.58	2087.13	1196.08	896.81	99.18	599.30	592.46	297.24

La tabla 5 muestra los resultados experimentales obtenidos bajo las condiciones anteriormente descritas. Cabe señalar que se codificaron 226 Frames para el codec secuencial y 14 Gops para el codec paralelo, donde cada GOP contiene 16 Frames, por lo que se codificaron 224 Frames (No. de frames = Fr). La diferencia del número

de Frames tiene que ver con la limitante en el patrón de codificación antes citada del algoritmo paralelo. Dicha versión tiene otra limitante, se tuvieron que establecer las pruebas con 14 GOPs, debido a que no se hace una codificación cíclica con lo cual el número GOPs depende del *tamaño* de la secuencia. En cuanto a los tiempos obtenidos y mostrados en la tabla 5 cabe subrayar que conforme se tiene mayor número de procesadores es menor el tiempo de codificación; sin embargo, esto no siempre se cumple entre 8 , 10, y 12 procesadores. Esto se debe a la forma en que se realizó la distribución de la carga para esta versión paralela.

En las figuras 2 y 3, se muestra la distribución de GOPs para 8 y 10 procesadores respectivamente, las cuales ilustran el *balance de la carga*. Se puede observar que con 10 procesadores se tienen más *tiempos muertos* que con 8. Si se hiciera el reparto de GOPs para 12 se harían evidentes aun más *tiempos ociosos* de los procesadores. Debido a este esquema de paralelización, se puede deducir que el tiempo de codificación no siempre se decrementa a pesar de ir incrementando el número de procesadores, ya que la distribución de GOPs esta predeterminada desde el inicio, con lo cual no se aprovechan los procesadores que vayan siendo liberados; retrasando así el *tiempo de codificación final*. Por esa misma razón el tiempo de codificación para 14 procesadores decrementa considerablemente, debido a que se tiene el mismo número GOPs con lo cual no se tiene ningún *tiempo muerto*. Cabe señalar que el tiempo de codificación que se toma es el del último procesador que termina el trabajo.

GOPS	1	3	5	7	9	11		
	0	2	4	6	8	10	12	13
P	1	2	3	4	5	6	7	8

Fig. 2: Distribución de 14 GOPs en 8 procesadores

GOPS	1	3	5	7						
	0	2	4	6	8	9	10	11	12	13
P	1	2	3	4	5	6	7	8	9	10

Fig. 3: Distribución de 14 GOPs en 10 procesadores

Tabla 6. FrameRate con 14 Gops calculado con la siguiente fórmula: Fr = Nf / Tp

Secuencia	2P	4P	4P	8P	10P	12P	14P
foreman_qcif	1.52	2.66	3.54	5.30	5.29	5.33	10.63
students_qcif	1.51	2.67	3.55	5.34	5.34	5.36	10.64
foreman_cif	0.37	0.66	0.87	1.31	1.31	1.30	2.61
students_cif	0.38	0.67	0.88	1.33	1.32	1.33	2.63
martin_720x480	0.11	0.19	0.25	0.37	0.37	0.38	0.75
ayersroc_720x480	0.11	0.19	0.25	0.37	0.37	0.38	0.75

Como se puede observar en la tabla 6, cada vez que se incrementa el número de procesadores el FramaRate incrementa más o menos en la misma proporción, no así

cuando se tienen 8, 10 y 12 procesadores. Esto es por lo mismo que se mencionaba con respecto a los tiempos.

El *FrameRate natural* o *tiempo real* de una secuencia de video es de 30 fps. Se puede observar en la figura 4 que con 14 procesadores se esta muy lejos de poder alcanzar ese número de frames codificados por segundo y que conforme se tiene mayor resolución más lejos se está de tener tiempo real. Dicho tiempo para codificar 14 GOPs es de 7 segundos, lo cual quiere decir que para comprimir 14 GOPs de la secuencia *foreman* en su formato *cif* se necesitan 37.5 procesadores y para *ayersroc_720x480* se requieren 130.6 procesadores.

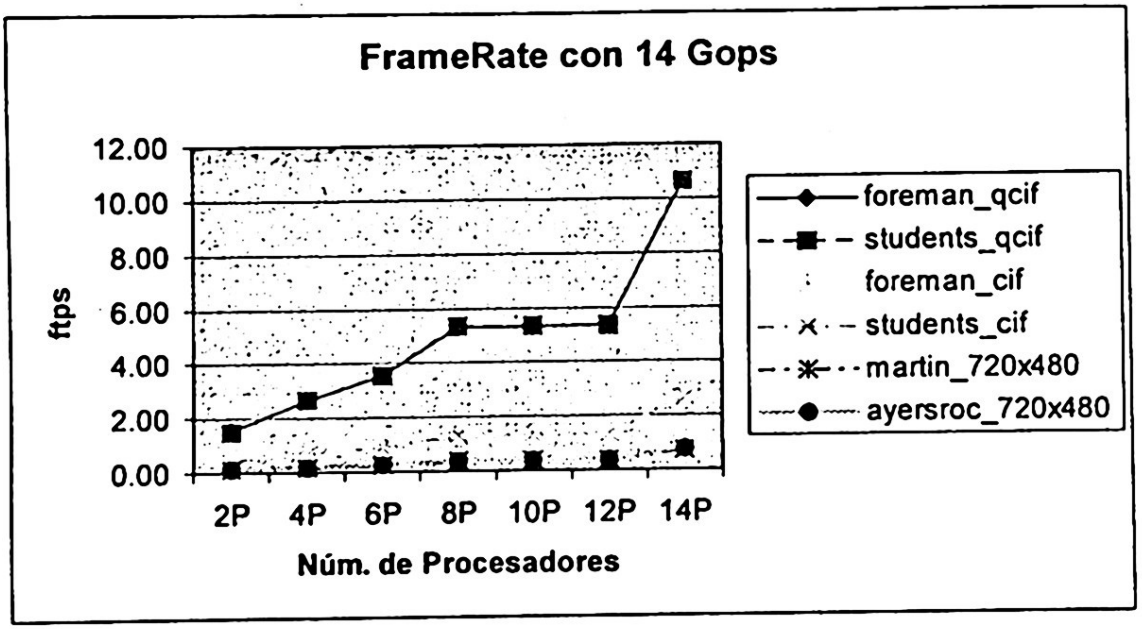

Fig. 4: FrameRate con 14 Gops

En la tabla 7 se muestra el SpeedUp (*aceleración*) de cada una de las secuencias para los números de procesadores empleados. Al igual que en los tiempos de codificación mostrados en la tabla 5 y su FrameRate expuesto en la tabla 6, con los procesadores 8, 10 y 12, el SpeedUp en ocasiones, en vez de aumentar, disminuye. Esto se debe a la misma causa, los tiempos muertos de los procesadores es mayor para 12 que para 10 y de éste que para 8.

Tabla 7. SpeedUp con 14 Gops calculado con la siguiente fórmula: $Sp = Ts / Tp$

Secuencia	Ts	2P	4P	6P	8P	10P	12P	14P
foreman_qcif	306.12	2.08	3.63	4.84	7.24	7.23	7.28	14.52
students_qcif	306.94	2.07	3.66	4.87	7.32	7.31	7.34	14.58
foreman_cif	1258.00	2.10	3.68	4.89	7.33	7.34	7.31	14.66
students_cif	1218.12	2.06	3.62	4.78	7.21	7.19	7.24	14.32
martin_720x480	4216.10	2.05	3.52	4.71	6.97	7.02	7.08	14.05
ayersroc_720x480	4268.58	2.05	3.57	4.76	7.12	7.12	7.20	14.36

codificar, no existen tiempos ociosos para ninguno de lo procesadores en ningún momento.

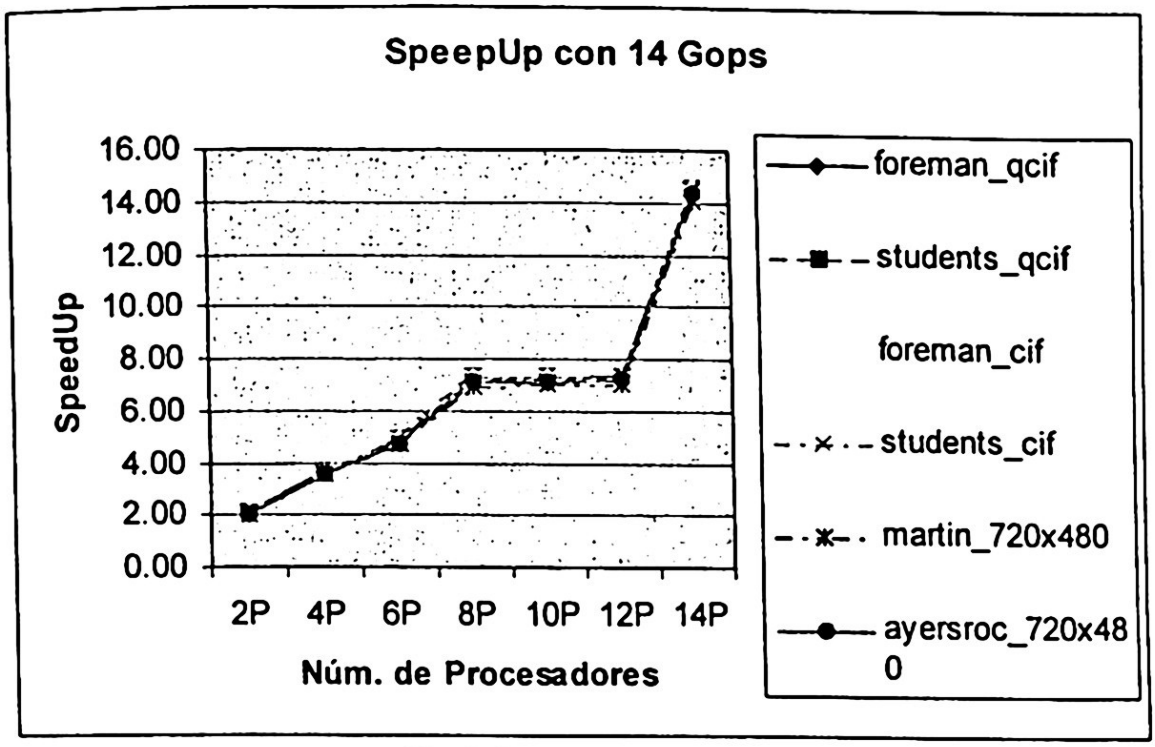

Fig. 5. SpeedUp con 14 Gops

6 Trabajo futuro

Como ya se había mencionado anteriormente, las pruebas mostradas en este capítulo establecen la base de una investigación con la finalidad de realizar una versión paralela que permita mejorar lo hecho. El objetivo es *implementar y evaluar un algoritmo paralelo con balanceo de carga* para la compresión de secuencias de video utilizando el codificador H.264, buscando reducir los tiempos de codificación bajo el *patrón idóneo* ya antes referido. Además se busca poder establecer una *codificación cíclica* a fin de que el número de frames no sea una limitante para realizar pruebas de codificación con mayor número de GOPs. También se espera determinar el tipo de video más adecuado para el codec a implementar.

El balanceo de la carga será por demanda, similar al planteado en el codec A3-frontera [4]. Su esquema de distribución consiste en que el proceso 0 (maestro) es el encargado de distribuir los identificadores de Gops a los procesadores conforme van desocupándose. El proceso maestro sabe que se ha desocupado un procesador porque cada uno de los procesos codificadores le informa que ha terminado su tarea.

Referencias

1. Cuesta, S. B., Gonzáles, I. F.: Paralelización del estándar de codificación de video H264.

2. Effelsberg, W., Steinmetz, R.: Video Compression Techniques (heidelberg: dpunkt.verlag). 1998
3. Hernández, S. J.: Análisis Comparativo de los Modelos de Programación Distribuida, Utilizando Cluster de Estaciones de Trabajo. Tesis de Maestría, Universidad Mexicana Plantel Veracruz. Agosto del 2001
4. Rodríguez, L. A.: Diseño e implementación de algoritmos paralelos para la compresión de secuencias de video MPEG4. Reporte Técnico. Universidad Politécnica de Valencia. Noviembre del 2002.