DISEÑO DE UN ALGORITMO PARALELO PARA CODIFICACIÓN DE VIDEO MPEG4, SOBRE UN CLUSTER DE COMPUTADORAS PERSONALES.

Autores:

Abelardo Rodríguez L.*, Manuel Pérez M. **, Alberto González T. **, José Hernández S.*

**Instituto Tecnológico de Veracruz*, México

***Universidad Politécnica de Valencia*, España.

E-mail: arleon@itver.edu.mx, mperez@disca.upv.es, agt@disca.upv.es, jhsilva@itver.edu.mx

Resumen.

Una de las razones de que no es tan común la comunicación basada en video a través de Internet, es porque el video requiere un ancho de banda considerable para ser trasmitido por este medio. Una alternativa es ampliar el ancho de banda en los canales de comunicación, lo cual aun con los avances logrados en recientes años, resulta sumamente costoso. La otra alternativa es comprimir el video para poder enviarlo a través de la red. Esta ultima alternativa es la que más se ha usado y ha dado mejores resultados. Los buenos resultados en la compresión se deben a que se han desarrollados sofisticados algoritmos de compresión de video como MPEG o H323 que minimizan la redundancia en el video. Sin embargo estos algoritmos de compresión requieren mucho tiempo para codificar un flujo de video, haciendo imposible la transmisión en tiempo real. Este articulo trata sobre una alternativa que se esta desarrollando para obtener un algoritmo de compresión de video que aproveche el procesamiento en paralelo para reducir el tiempo de compresión y hacerlo tender a ser igual tiempo de generación de video, es decir lograr codificar a la misma velocidad que se recibe el video, con lo cual se logrará codificación cercana a tiempo real.

I. Introducción

Las tecnologías para el manejo de información han evolucionado rápidamente, dando lugar nuevos sistemas, tales como los de video conferencias que permiten distribuir flujos de video mediante una red. Estos sistemas suelen ser caros, ya que requieren infraestructura especial, tal como conectividad de banda ancha y computadoras de alto desempeño [1].

Sin embargo, el tener un ancho de banda amplio no es el principal problema, ya que enviar flujos de video digitalizado, resulta muy costoso. La solución sigue siendo comprimir el video antes de enviarlo para que al recibirlos los clientes los descompriman y puedan usarlo. Actualmente este proceso de compresión en línea (en el momento que se recibe en video, comprimirlo y enviarlo) requiere de mucha capacidad de computo, las cuales resultan caras y poco accesibles al usuario común.

En cuanto al formato de compresión de video existen varios estándares que actualmente se usan, tales como H323 y MPEG. De estos dos, MPEG[2] parece ser el que más futuro promete, ya que actualmente considera además extensiones para manipulación de elementos multimedia dentro de los flujos de video, con lo cual esta -al menos- un paso adelante de otros estándares que solo consideran el manejo de video.

2. Problemática:

La eficiencia de los algoritmos exitosos de compresión (como MPEG4 por ejemplo) se basa especialmente en la eliminación de la redundancia de la información[2] que contiene las imágenes que se van a comprimir. Esta eliminación va enfocada en dos aspectos principalmente.

La redundancia espacial. Toma como información de entrada una maya de píxeles (frame), en la cual se reduce la información redundante (como los fondos de áreas grandes de la imagen), usando algoritmos de compresión, como el JPEG.

La redundancia temporal. Consiste en tomar en cuenta que mucha de la información de un frame se repite también en los siguientes, con lo cual se puede reducir mas información, si se codifica solo los cambios con respecto al anterior. Esto se logra a través de Algoritmos de Estimación de Movimiento que calculan la trayectoria que seguirá la parte del frame que no cambia.

Estas técnicas de compresión son muy efectivas ya que pueden ahorrar hasta un 90% de espacio, sin embargo requieren de mucho tiempo para llevar a cabo este proceso tan complejo.

Estos algoritmos de compresión, están pensados para dar excelentes resultados en el tamaño de video codificado, pero sin considerar el tiempo de compresión requerido.

Como el proceso de codificación suele requerir mucho poder de computo generalmente se obtiene usando computadoras superescalares que ayuden con su "fuerza bruta" computacional a llevar a cabo este proceso, con tiempos de retardo aceptables.

Una alternativa que se ha empezado a estudiar y que no requiere costosas computadoras, es usar cluster de estaciones de trabajo [3], las cuales abundan en todos los lugares y que además en la mayoría de los casos son subutilizadas.

Para aprovechar dichos clusters hay que implementar los algoritmos de compresión (en este caso el MPEG-4) usando programación distribuida. Para ello algunos autores han propuesto diversas alternativas de solución como por ejemplo segmentar el flujo de video en secciones que permitan dedicar cada segmento del video a varios procesadores [6].

En el ámbito del aprovechamiento del poder de procesamiento que proporcionan un cluster, existen varias alternativas, siendo MPI una de las mas recientes y completas, ya que engloba las características muchos esquemas de programación anteriores[5].

3. Desarrollo:

Para desarrollar una versión paralela del codec MPEG4 que corriera en forma distribuida en un cluster de estaciones de trabajo, se usó WMPI, una versión comercial de la librería estándar MPI para paso de mensajes. El modelo de programación usado es el de esclavo maestro, ya que es un modelo diferente al usado en otros trabajos, el cual resulta simple de implementar en este trabajo inicial, buscando ver que tanto aporta, a este proceso de codificación.

El código fuente del codec MPEG4 (fdam) fue desarrollado en Visual C++ por un grupo de investigadores patrocinado por Microsoft [7], basado en el ISO/IEC JTC 1/SC 29 [8].

Dicho codec usa como flujo de entrada (video) el formato YUV 4.2.0. Para las pruebas de la versión paralela se uso como entrada una secuencia de video de prueba estándar de 10 segundos, llamada "Mother_and_Daugther" que esta en formato qcif (176x144) a 30 fps, en formato YUV4.2.0.

Dicho flujo se dividió en Gops (Group Of Pictures) de 1 segundo cada uno. Estos Gops fueron enviados a cada uno de los procesos participantes.

El cluster usado para las primeras pruebas esta compuesto de 8 estaciones de trabajo con Windows NT, conectadas con una red Fast Ethernet de 100 Mb y cada estación tiene un procesador Pentium III de 866 Mhz .

El programa consta esencialmente de dos tipos de procesos. El “maestro” que lee el flujo de video y lo distribuye y los procesos “esclavos” que se encargan cada uno de ellos de codificar una porción del flujo. Una vez terminado el proceso de codificación en cada estación “esclavo”, es regresado con los resultados al proceso “maestro” que los recibe y conjunta.

Se hicieron dos tipos de corridas. La primera fue usando todos los procesos MPI en la misma maquina, es decir cada proceso era un thread en el mismo procesador. Esta prueba se puede considerar como la testigo, equivalente al procesamiento secuencial. Este arreglo de procesos se ilustra con en la figura 1.

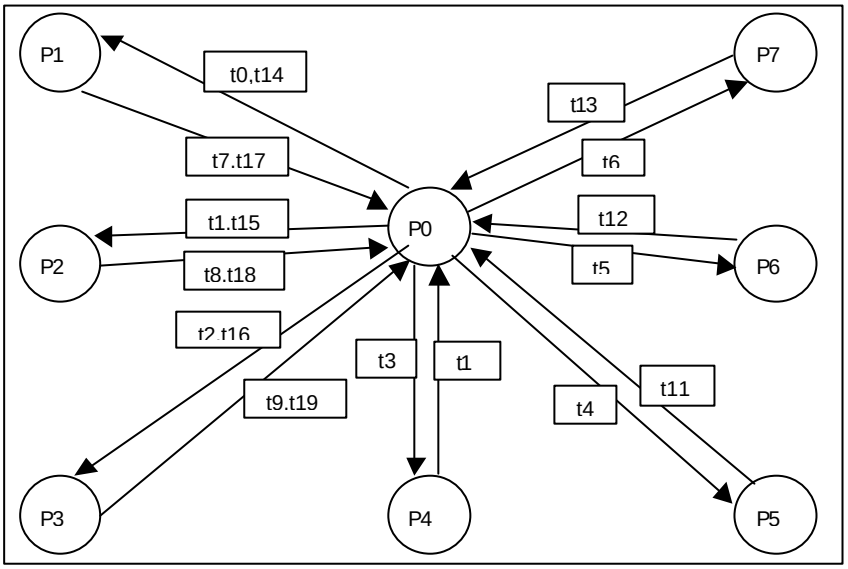

Figura 1: 8 procesos en un solo procesador

En la segunda prueba, los 8 procesos (el maestro y 7 esclavos) están corriendo cada uno en una maquina. De esta forma se obtiene ya una ventaja significativa en la reducción del tiempo total de procesamiento, ya que cada computadora aporta su capacidad de computo al proceso general. Este arreglo de procesos se ilustra en la Figura 2.

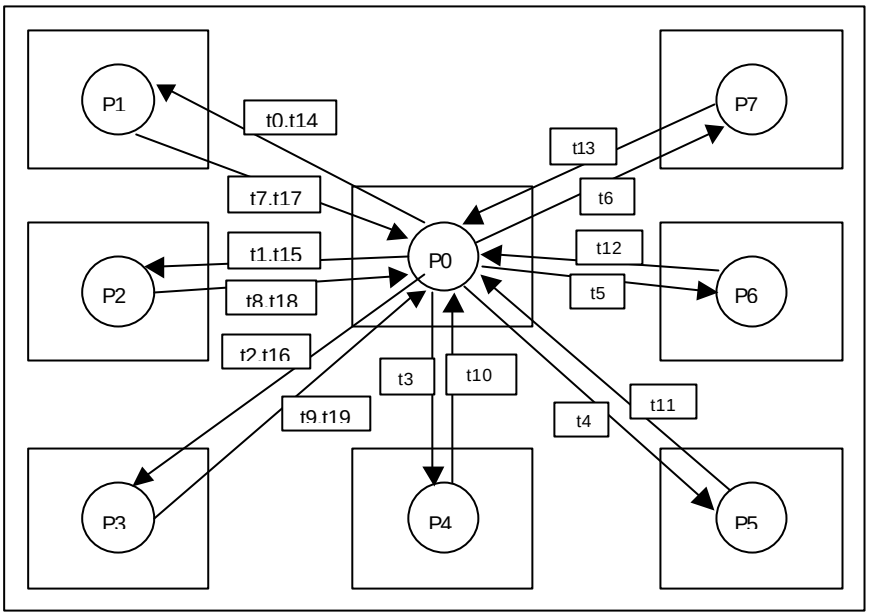

Figura 2: 8 procesos en 8 procesadores

Los estados de los dos casos se pueden observar en la tabla 1. Las letras "P" representan a los procesos, que van de 0 a 7. Las letras "t" representan los estados. En el caso de esta primera implementación son 20 estados ya que 10 son para enviar GOP y 10 para recibir. Por ello van de t0 a t19.

Estados	Descripción
t0	P0 envía Gop 0 a P1.
t1	P0 envía Gop 1 a P2.
t2	P0 envía Gop 2 a P3.
t3	P0 envía Gop 3 a P4.
t4	P0 envía Gop4 a P5.
t5	P0 envía Gop 5 a P6.
t6	P0 envía Gop 6 a P7.
t7	P1 envía Gop 0 a P0.
t8	P2 envía Gop 1 a P0.
t9	P3 envía Gop 2 a P0.
t10	P4 envía Gop 3 a P0.
t11	P5 envía Gop 4 a P0.
t12	P6 envía Gop 5 a P0.
t13	P7 envía Gop 6 a P0.
t14	P0 envía Gop 7 a P1.
t15	P0 envía Gop 8 a P2.
t16	P0 envía Gop 9 a P3.
t17	P1 envía Gop 7 a P0.
t18	P2 envía Gop 8 a P0.
t19	P3 envía Gop 9 a P0.

Tabla 1: Descripción de los estados en los dos casos de prueba.

4. Resultado:

El tamaño de cada GOP sin codificar es de 1140480 bytes, el cual es fijo. Ya una vez codificados los GOPS, el tamaño en promedio es de 10524 bytes.

Los tiempos para las ejecuciones de los programas en el arreglo secuencial, se pueden observar en la tabla 2 en la columna Prueba1.

	Prueba 1			Prueba 2		
No. GOP	T. Rec	T.Codif.	T. G. Codif.	T. Rec	T.Codif.	T. G. Codif.
0	320	19648	6817	180	19458	6817
1	231	27569	9959	6769	27719	9959
2	191	27770	11077	4356	29002	11077
3	190	26989	14785	9533	28091	14785
4	310	27570	12997	34910	28692	12997
5	190	30294	12134	31004	31295	12134
6	180	29282	11523	724722	29523	11523
7	181	22412	11728	14061	22432	11728
8	190	23845	7543	5789	23854	7543
9	180	19228	6685	4486	20480	6685
	T.. Total		**257381**			**57753**

Tabla 2: Resultados de codificación secuencial y paralela

En la prueba 2 el programa aprovechar las 8 maquina, atendiendo cada una a 1 proceso, tal como se ve en la Figura 2. El programa codificado es el mismo que el caso anterior, sin embargo los tiempos son significativamente menores, como se pueden observar en la ultima fila de la tabla 2, en la columna Prueba2.

5. Discusión:

Inicialmente no se pueden hacer comparaciones entre el modelo desarrollado en este trabajo (esclavo - maestro) y los resultados obtenidos en otros trabajos, ya que se tendría que replicar la solución de otros en condiciones similares, lo cual no se ha hecho en este trabajo. Por ello se presentan solo los resultados de la versión paralela, comparándolos contra la versión secuencial del codec.

Con respecto al tamaño de los Gops codificados se obtuvo una reducción del 92% del tamaño original. Este resultado es el esperado, ya que se sabe de la eficiencia del algoritmo MPEG4 para eliminar redundancia. Sin embargo como se sabe también, el precio es alto ya que para llevar a cabo esta reducción, se requiere demasiado poder de computo, el cual si no se tiene se traduce en tiempo adicional en el procesamiento[3].

Con respecto a los tiempos de codificación del arreglo secuencial, que se observaron en la tabla 2, son diferentes, porque como se sabe el proceso de codificación esta en relación con el contenido del flujo de video, el cual es variable. Por ello algunos flujos (GOPs) se llevan mas tiempo que otros para ser codificados.

Al final de la tabla 2 aparece un renglón que dice Tiempo Total del Proceso. Este tiempo fue tomado desde el inicio de la distribución de GOPs, hasta el final de la recepción del último GOP codificado. Como se puede observar el tiempo para la Prueba 1 es de 257381 milisegundos es decir 257.38 seg o 4.28 minutos, esto para codificar 10 segundos de video. Para la codificación de la prueba 2 el tiempo fue de 57783 milisegundos es decir 57.783 segundos. Lo anterior quiere decir que el tiempo de compresión se redujo de 257 segundos a 57 segundos, en términos porcentuales se redujo a un 22% del tiempo de compresión, usando 8 procesadores.

6. Conclusiones

Como se puede observar la reducción de tiempo es significativa con 8 procesadores, sin embargo no es proporcional al numero de procesadores. Es decir, si la prueba secuencial se llevó 257 segundos en un solo procesador, al agregar 8 maquinas debimos haber tenido un tiempo de 32 segundos de codificación y no del 57. La diferencia es porque el algoritmo no esta del todo optimizado aun y por las perdidas que ocasiona usar una red de área local como medio de comunicación. Para reducir mas los tiempos se trabajará en tres vertientes: Optimizar el algoritmo paralelo, aumentar el numero de procesadores participantes y mejorar el medio de comunicación.

En la optimización hay mucho que realizar y es donde irán encaminados los primeros esfuerzos, agregando un planificador de tareas y evitando las perdidas de tiempo en tareas redundantes dentro del codificador MPEG4, esto se hará usando dos niveles de paralelización.

En la vertiente de agregar mas maquinas, algunos autores [4] han comprobado que hay un punto en el cual el número de procesadores no contribuye significativamente a mejorar los resultados, por lo que la solución no es definitiva.

En el aspecto de mejorar la red, se piensa manejar un Middleware más sólido y de alta velocidad, por ejemplo usar una red de mejor velocidad como Myrinet o incluso procesadores fuertemente acoplados.

Se seguirá trabajando en este campo hasta obtener una codificación de video de alta calidad, teniendo al menos un tiempo igual al que dura el video. Esta es la siguiente meta de esta investigación.

Bibliografía:

[1] Orne las Ceja, Eduardo; et all; Tesis de Ingenieria; http://video.comserv.ipn.mx/

[2] Rob Koenen; ***"International Organisation For Standardisation Organisation Internationale de Normalisation ISO/Iec JTC1/Sc29/Wg11 coding of Moving Pictures And Audio"***; http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm, Marzo del 2001.

[3] Olivares Montes, Teresa. ***"Codificación de Video MEPG-2 sobre una red de estaciones de trabajo"***. Depto. de Informática, Universidad de Castilla-La Mancha.

[4] Anastasios, et all. ***"Concurrency Analysis for Real Time MPEG-4 Video Encoding"***. Articulo Departamento de Ciencias Computacionales, Universidad de Londres.

[5] Peter Pacheco, ***"A User´s guide to MPI"***. Departamento de Matemáticas de la Universidad de San Francisco. Marzo de 1998.

[6] Akramullah S. M.. ***"Real Time MPEG2 Video Encoding on Parallel and Distributed Systems"***. Tesis de Maestría en Filosofía en Ingeniería Eléctrica y Electrónica de la Universidad Hong Kong de Ciencia y Tecnológia. Julio de 1995

[7] Microsoft Corp. ***"Microsoft MPEG-4 Visual Reference Software"***. Julio 3 del 2000 FDAM1-2.3-001213, Versión 2.3.0

[8] International Standard Organization. ***"ISO/IEC JTC 1/SC 29/WG 11 -Coding of Moving Pictures and Audio"***. Enero 31 del 2001.