Dix-septième colloque GRETSI, Vannes, 13-17 septembre 1999

Applications vidéo sur des systèmes à base de processeurs généralistes : L'exemple de l'encodeur MPEG2

Eric DEBES, Fulvio MOSCHETTI

Laboratoire de Traitement des Signaux,
Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne

Eric.Debes@epfl.ch, Fulvio.Moschetti@epfl.ch

Résumé – Cet article explique comment des logiciels multimédia qui nécessitent beaucoup de ressources peuvent s'exécuter de manière satisfaisante sur des stations de travail à base de processeurs généralistes, sans matériel spécifique, grâce à des changements algorithmiques et à l'adaptation de l'implémentation aux architectures PCs. L'encodeur MPEG2 est pris comme exemple pour décrire la méthodologie qui a pour but d'exécuter l'application le plus rapidement possible sur une station de travail PC. Afin d'atteindre cet objectif, nous montrons comment obtenir les gains en vitesse grâce à une technique de comparaison de blocs différente de celle de la recherche exhaustive et grâce à l'exploitation des différents niveaux de parallélisme disponibles dans les machines à base de processeurs généralistes. Les résultats démontrent un gain considérable en vitesse tout en conservant une qualité acceptable du bitstream vidéo décodé.

Abstract – This paper explains how very resource-demanding multimedia softwares can run satisfactorily fast on General Purpose Processor based workstations, without additional hardware, thanks to changes at the algorithm level and to the adaptation of the usual implementation to PC architectures. An MPEG2 encoder is taken as an example to describe the methodology which aims at letting the application run as fast as possible on a PC workstation. In order to reach this objective we show how the speed improvement can be obtained using a Block Matching different from Full Search and exploiting the different levels of parallelism that can be found in general purpose machines. The results show considerable gain in speed, while maintaining a good video quality.

1. Introduction

Dans le domaine du traitement des images et de la vidéo, l'implémentation des algorithmes est très souvent réalisée à l'aide de matériel spécifique ou de DSPs. Aujourd'hui les processeurs généralistes utilisés dans les PC sont devenus très puissants et, grâce à des volumes de production importants, bon marché. De plus, même si les processeurs généralistes n'ont pas été développés pour les applications vidéos, ils ont évolué en intégrant la technologie SIMD (Single Instruction Multiple Data) et ils sont sur le point d'adopter la technologie de parallélisme explicite VLIW (Very Long Instruction Word). Grâce à ces évolutions, les applications vidéos peuvent tourner à des vitesses "considérables" aujourd'hui sur ces machines. Outre leur puissance, les processeurs généralistes ont l'avantage de pouvoir être utilisés pour différentes applications alors que les DSPs sont spécifiques à chaque application.

Au niveau des logiciels multimedia, les applications les plus difficiles à gérer du point de vue ressources de calcul sont les applications vidéos et en particulier l'encodage video à partir de séquences d'images acquises par une caméra ou même déjà disponibles sur un disque. Dans cet article, l'encodeur MPEG2 est pris comme exemple représentatif d'un grand nombre d'applications pour montrer comment adapter les algorithmes à l'architecture et en exploitant les différents niveaux de parallélisme disponibles dans les systèmes à base de processeurs généralistes. Dans la section 2, nous présentons l'algorithme d'estimation de mouvement. Dans la section 3, nous décrivons la méthodologie utilisée pour exploiter les différents niveaux de parallélisme disponibles dans les systèmes à base de processeurs généralistes. Dans la section 4, nous présentons les résultats obtenus en terme d'augmentation de vitesse d'exécution et nous proposons d'autres applications de cette méthodologie. Les conclusions sont données dans la section 5.

2. Estimation de Mouvement

Dans cette partie, un nouvel algorithme d'estimation de mouvement est présenté en remplacement de celui qui effectue une recherche exhaustive utilisé dans l'implémentation du Software Simulation Group (SSG) [1] lors du développement du standard MPEG2.

2.1 Détection des goulets d'étranglement

Dans un premier temps, le logiciel VTune [2] d'Intel nous a permis de détecter les fonctions qui nécessitent les temps de calcul les plus importants. Certaines de ces fonctions étaient bien sur connues au départ (la routine d'estimation de mouvement, la DCT, etc…), mais VTune donne exactement le pourcentage de temps passé par le processeur dans chaque fonction. Dans la version du SSG, le processeur passe plus de 90% de son temps dans la fonction d'estimation de

mouvement, d'où la nécessité d'améliorer cette fonction au niveau algorithmique.

2.2 Présentation d'un nouvel algorithme d'estimation de mouvement

L'estimation de mouvement la plus utilisée dans la compression de vidéo numérique est la comparaison de blocs. Dans cette méthode, le mouvement est estimé bloc par bloc, c'est-à-dire que l'image à encoder est divisée en blocs de tailles égales et le vecteur de mouvement pour chaque bloc est estimé en cherchant le maximum de corrélation avec un bloc dans une fenêtre «dite de recherche» dans l'image de référence.

Il existe un grand nombre de méthodes pour réaliser la comparaison de blocs. L'algorithme de recherche complète (RC) ou recherche exhaustive, par exemple, compare chaque bloc de l'image à coder à tous les blocs de la fenêtre de recherche dans l'image de référence. Un autre algorithme très connu est l'algorithme à 3 pas (R3P) [3]. C'est l'un des plus rapide disponible dans la littérature. Une analyse complète des techniques classiques d'estimation de mouvement est donnée dans [4].

Dans le cadre du travail présenté dans cet article, un algorithme de recherche hybride (RH) a été développé dans lequel la première recherche se fait sur une grille dont les pixels sont éloignés de 4 pixels et uniformément répartis à partir du centre de la fenêtre de recherche. La deuxième étape se fait sur les 8 sommets éloignés de 2 pixels du minimum obtenu lors de la première étape et de même pour la troisième étape avec une distance d'un seul pixel (cf. Fig. 1).

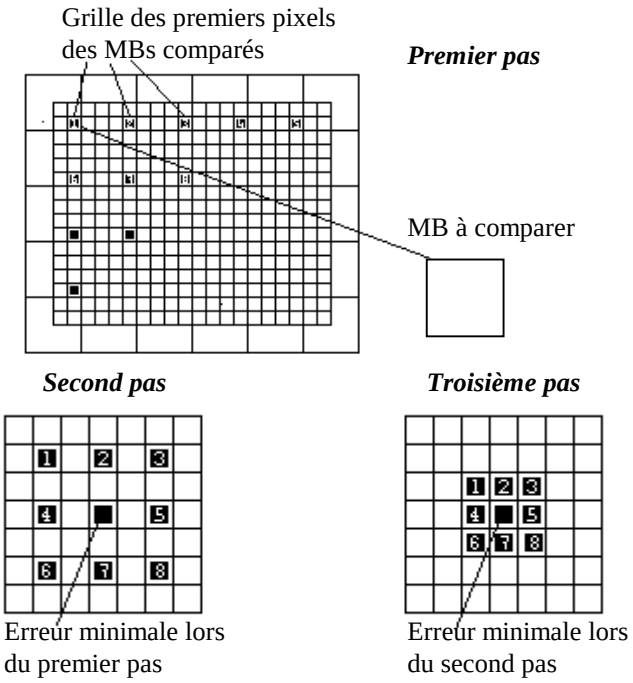

FIG. 1 : Description des 3 pas de l'algorithme de recherche hybride

2.3 Comparaison avec d'autres algorithmes classiques

Le tableau 1 montre le nombre d'opérations requises par les différents algorithmes pour une fenêtre de recherche de largeur 2L.

TAB. 1 : Nombre d'opérations pour les divers algorithmes

Algorithme	Nombre de comparaisons	L =16	Temps*
RC	$(2L+1)^2$	1089	24'47"
R3P	$1+8\log_2 L$	33	5'16"
RH	$(L/2+1)^2 + 16$	97	6'26"

La figure 2 donne les valeurs du PSNR pour la séquence basket encodée à 5Mbits/s.

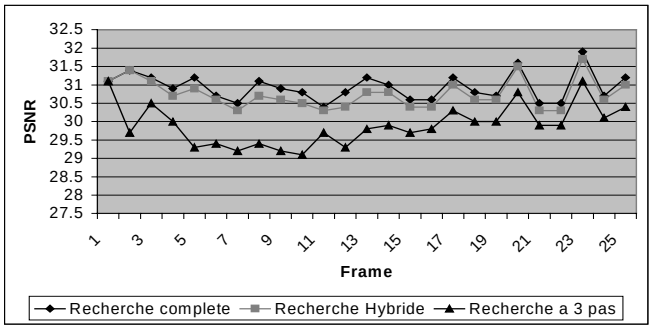

FIG. 2 : PSNR pour chaque type de recherche

Ces deux figures montrent que, tout en étant quatre fois plus rapide que RC et à peine plus lente que R3P, la RH donne une qualité très proche de celle de la RC. Le but étant d'accélérer l'exécution de l'encodeur sans trop affecter la qualité, cet algorithme convient tout à fait à notre utilisation.

La RH présente aussi l'avantage de s'adapter à la largeur de la fenêtre de recherche contrairement à la R3P qui va donner des résultats de plus en plus mauvais pour une fenêtre de recherche plus grande. La RH s'adapte à des séquences très différentes avec beaucoup ou peu de mouvement.

En utilisant une nouvelle fois le logiciel VTune d'Intel, le temps passé par le processeur dans la routine d'estimation de mouvement est maintenant descendu de 90% à environ 60% et l'encodeur tourne 3,85 fois plus rapidement que la version du SSG.

3. Exploitation des différents niveaux de parallélisme

Dans cette partie, les différents niveaux de parallélisme disponibles dans les systèmes à base de processeurs généralistes sont présentés et leur efficacité pour les applications vidéos est discutée. Pour chacun des niveaux, le gain potentiel est étudié et le gain pour l'encodeur MPEG2 obtenu sur un HP Kayak XU équipé de deux processeurs Pentium II à 300MHz est présenté.

3.1 Parallélisme de données et instructions multimédia

La technologie SIMD a été ajoutée aux processeurs généralistes afin d'accélérer l'exécution d'applications intensives qui traitent des données entières de petites tailles (typiquement 8 bits) avec de nombreuses opérations parallèles. La technologie MMX d'Intel par exemple consiste en une séries d'instructions qui s'exécutent en parallèle sur des paquets de 8 ou 16 bits à l'intérieur de registres de 64 bits.

La technologie SIMD représente bien entendu un niveau de parallélisme de choix pour les applications vidéo. Grâce aux instructions MMX le gain idéal est 8 lorsque les pixels sont codés sur 8 bits, car 8 instructions peuvent être exécutées en parallèle. Ce gain n'est malheureusement jamais atteint, car toutes les instructions ne peuvent pas être exécutées en parallèle et les données doivent être passées des registres MMX aux registres entiers lorsque l'opération est terminée. Pour le calcul de la somme des valeurs absolues de la différence entre deux blocs, le gain en vitesse obtenu est d'environ 5. En appliquant la même méthodologie à d'autres fonctions il a été possible d'accélérer l'exécution de l'encodage d'un facteur 3,9 grâce à cette technique, qui cumulé à l'optimisation algorithmique permet à l'encodage de s'effectuer 14,4 fois plus vite que la version du SSG.

Le désavantage de cette technique réside dans le fait qu'aucun compilateur ne produit de code assembleur avec des instructions MMX et que la plupart des librairies existantes ne sont pas vraiment efficaces, car elles nécessitent souvent une conversion des données. Le meilleur moyen d'obtenir une implémentation efficiente consiste à écrire directement les principales fonctions en assembleur.

3.2 Parallélisme dans les processeurs superscalaires

Les processeurs superscalaires [5] disposent de plusieurs unités d'exécution qui permettent aux instructions de s'exécuter en parallèle. Dans les processeurs comme le Pentium, par exemple il existe deux pipelines U et V dans lesquels peuvent être placées des instructions sur des entiers. En règle générale, il est nécessaire de changer l'ordre des instructions afin que les instructions soient «pairables» dans les deux pipelines. Dans la génération des Pentium Pro et PentiumII-III, l'exécution se fait indépendamment de l'ordre des instructions («out-of-order execution») et les résultats sont rassemblés à la sortie du pipeline. Dans ce cas il suffit de suivre quelques règles données dans les manuels de programmation et d'utiliser des compilateurs optimisés pour l'architecture considérée afin d'obtenir des gains en vitesse intéressants.

3.3 Parallélisme des instructions

Ce type de parallélisme est comparable à la technologie SIMD qui peut être considérée comme du parallélisme à l'intérieur du mot de données alors que la technologie VLIW [6] correspond à un parallélisme à l'intérieur du mot d'instruction. Dans un processeur VLIW, un mot d'instruction contient plusieurs instructions qui sont décodées puis exécutées en parallèle par les différentes unités d'exécution. Contrairement à de nombreuses architectures superscalaires, dans lesquelles le processeur décide durant l'exécution dans quelle unité de traitement chaque instruction va être exécutée, dans une architecture VLIW, c'est le compilateur qui décide où chaque partie du mot d'instruction sera exécutée.

Cette technique sera implémentée dans la future génération de processeurs développée par Hewlett-Packard et Intel IA-64 dont le premier élément a pour nom de code «Merced» [7], dans lequel trois instructions pourront être exécutées en parallèle à l'aide d'un seul mot d'instruction. Grâce à un grand nombre d'unités d'exécution et à la technique des prédicats, ce processeur permettra d'obtenir des gains en vitesse très importants pour les algorithmes de traitement vidéo, tout à fait adaptés à ce genre de parallélisme.

L'efficacité de ce niveau de parallélisme dépend principalement du nombre d'instructions dans le VLIW ainsi que du nombre d'unités d'exécution dans le processeur. Il dépend aussi beaucoup de la qualité du compilateur ou, si le codage se fait directement en assembleur, de l'effort fait pour mettre le maximum d'instructions dans un VLIW. A cause de ces différents facteurs il est très difficile de prédire l'efficacité de ce niveau de parallélisme pour des applications vidéos. Cependant en ce qui concerne la famille IA-64, on peut espérer obtenir un nombre moyen d'instructions par cycle légèrement supérieur à 2 (pour un maximum de trois).

3.4 Parallélisme multiprocesseur

Beaucoup de stations de travail actuelles disposent de plus qu'un processeur. Cependant, la plupart des applications qui tournent sur ces machines n'utilisent qu'un seul «thread» et, de ce fait, ne peuvent pas exploiter complètement la puissance de calcul de ces machines. Si une application avec un seul thread tourne sur une machine à deux processeurs, la charge de travail maximum sera 50% et, si aucune autre application ne tourne en même temps, le reste de la capacité de calcul sera perdue.

Par contre, en utilisant la technique «du multithreading» [8], il est possible de répartir la charge de travail sur les différents processeurs. En principe le gain qu'il est possible d'obtenir en parallélisant une application sur deux processeurs est de 2. Le but de l'effort de parallélisation est de créer suffisamment de threads pour pouvoir équilibrer la charge entre les processeurs. Cependant il faut noter que la gestion des threads, et plus particulièrement leur création et leur destruction, prend beaucoup de temps. C'est pourquoi il importe de créer le nombre de threads nécessaire et suffisant pour repartir la charge entre les différents processeurs.

Pour l'encodeur MPEG2, de nombreuses fonctions ont été parallélisées (estimation de mouvement, quantification directe et inverse, DCT et IDCT, …) et le gain final est supérieur à 1,8 sur une machine à deux processeurs.

Ce niveau de parallélisme peut s'étendre à des systèmes avec plus de deux processeurs. Malheureusement au moment où cet article a été rédigé, les processeurs PentiumII n'étaient disponible que pour des machines à deux processeurs.

Cependant une version de l'encodeur sans l'optimisation MMX a été testée sur un serveur avec 4 processeurs Intel Pentium Pro. Les résultats expérimentaux donnés dans la figure 3 montrent que l'encodeur s'adapte à une machine équipée de 4 processeurs avec un gain presque linéaire et supérieur à 3,2 pour 4 processeurs.

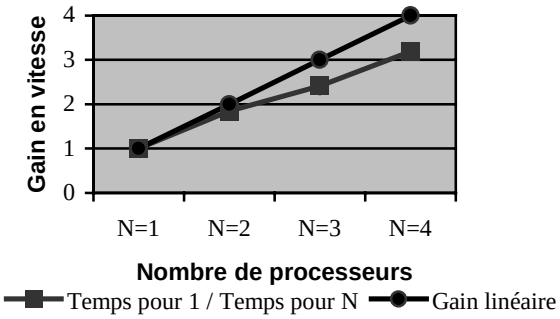

FIG. 3 : Gain en fonction du nombre de processeurs

4. Résultats et travaux futurs

Dans cette partie, les gains obtenus pour l'encodeur MPEG2 sont récapitulés et les travaux futurs sont présentés.

Les gains cumulés obtenus à chaque étape sont présentés dans la figure 4 par rapport à l'implémentation du SSG. Le nouvel algorithme d'estimation de mouvement présenté dans la seconde partie conduit à un gain de vitesse de 3,85. L'implémentation de certaines parties de l'estimation de mouvement, de certaines fonctions de quantification ainsi que de la DCT et le l'IDCT en assembleur à l'aide d'instructions MMX a permis de gagner un autre facteur de 3,9 en vitesse pour arriver à un encodeur qui tourne 14,4 fois plus vite que la version originale.

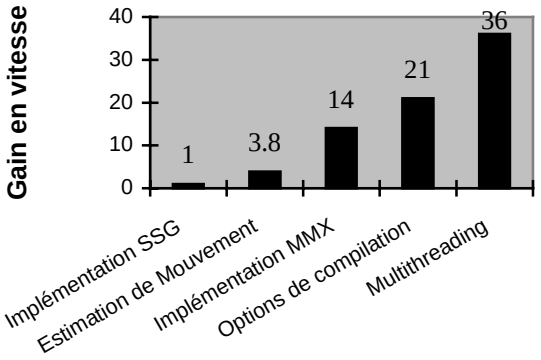

FIG. 4 : Récapitulation des gains obtenus

Puis l'utilisation de différents compilateurs (principalement Microsoft et Intel) avec différentes options d'optimisation pour adapter la compilation à l'architecture Intel PentiumII a permis de gagner encore 50% en vitesse. Pour finir le parallélisme multiprocesseur a été utilisé en parallélisant plusieurs routines afin d'exploiter les deux processeurs disponibles pour l'encodage. Le gain obtenu grâce à ce niveau de parallélisme est de 1,8.

Après ces différentes étapes, l'encodeur tourne 36,4 fois plus vite que la version du SSG. Les techniques utilisées ont déplacés le goulet d'étranglement du processeur vers la hiérarchie mémoire. L'utilisation des différents niveaux de parallélisme du système, et tout particulièrement les instructions multimédia, crée davantage de données en moins de cycles et augmente le trafic entre la mémoire cache et la mémoire RAM. C'est pourquoi l'un des objectifs est d'améliorer ce trafic à l'aide des nouvelles instructions de pré-chargement disponibles dans les PentiumIII.

Les futurs travaux vont aussi s'intéresser au parallélisme des instructions; les fonctions principales vont être parallélisées à ce niveau afin d'exploiter l'architecture des processeurs de la famille IA-64. D'autre part, dans cette famille, le grand nombre de registres généralistes disponibles permettra de diminuer beaucoup le trafic dans la hiérarchie mémoire.

5. Conclusions

Dans cet article nous avons démontré que la combinaison d'algorithmes optimisés pour les fonctions principales et de l'exploitation des différents niveaux de parallélisme disponibles dans des systèmes à base de processeurs généralistes permet d'atteindre des gains importants en vitesse pour des applications multimédia avec une perte en qualité très faible. L'application de ces différentes techniques a été présentée pour une application spécifique, mais a été validée avec d'autres applications telles que des encodeurs MPEG4 ou H.263. Avec les processeurs disponibles dans le futur, qui présenteront davantage de niveaux de parallélisme, comme par exemple les VLIW dans la future architecture IA-64, les applications multimédia tourneront encore plus vite sur les stations de travail de type PC.

Références

[1] MPEG2 Video Encoder, Version 1.2, July 1996, MPEG Software Simulation Group, http://www.mpeg.org

[2] Intel Corporation, VTune Performance Analyzer, http://developer.intel.com/vtune/index.htm

[3] M T.Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, *Motion compensated intraframe coding for video conferencing*, in Proc. NTC 81, pp C9.6.1-9.6.5, New Orleans, LA, Nov. 1981.

[4] P. Nesi, *Real Time Motion Estimation*, Real Time Imaging: Theory, techniques and applications. Laplante, Philip A., Stoyenko Alexander D. IEEE Computer Society Press 1996.

[5] M. Johnson, *Superscalar Microprocessor Design*, Prentice Hall, 1991.

[6] P. Faraboschi, G. Desoli, Joseph A. Fisher, *The Latest Word in Digital and Media Processing*, IEEE Signal Processing Magazine, March 1998.

[7] Intel Corporation, May 1999, IA-64 Application Developer's Architecture Guide.

[8] Hughes, *Object Oriented multithreading using C++*, Wiley Computer Publishing, 1997.