Application coopérative et gestion de l'énergie du processeur

Richard Urunuela
École des mines de nantes
projet Obasco EMN/INRIA
rurunuel@emn.fr

Résumé :
La puissance de calcul des systèmes mobiles rend possible l'exploitation d'application multimédia. C'est souvent au détriment de l'autonomie. Ce problème est d'autant plus complexe que l'élément clef, le processeur, est géré par l'ordonnanceur et le gestionnaire d'énergie. Notre solution repose sur une coopération entre les applications et le système pour réaliser les objectifs suivants, Maximiser la durée de vie des batteries et fournir la puissance de calcul nécessaire aux applications.

1. Introduction

Les applications multimédias sur des systèmes mobiles généralistes connaissent de plus en plus de succès auprès du grand public. Ces applications sont caractérisées par une consommation très importante en ressource processeur. Cette consommation de puissance de calcul entraîne une consommation électrique importante, qui diminue d'autant l'autonomie attendue par l'utilisateur. Le système doit gérer soigneusement l'énergie et le processeur. Cette gestion est particulièrement délicate pour les applications multimédias car le besoin en ressource processeur peut varier énormément d'une trame à une autre. De plus, les applications multimédias sont des applications temps réel molles pour lesquelles un retard d'exécution de quelques dixièmes de secondes a un impact sur la qualité du traitement. La plupart des gestionnaires d'énergie sur les systèmes d'exploitation généralistes ne sont pas adaptés à l'exécution d'applications avec des changements rapide de leurs besoins processeur. Cette inefficacité a souvent pour conséquence une gestion médiocre de la consommation électrique. Dans ce papier nous proposons une solution pour qu'une application coopère avec le système pour spécifier ses besoins énergétiques. De ce point de vue, le composant critique est le processeur de par l'ordonnanceur et de sa consommation électrique.

Gestionnaire de «DVS» et application multimédia

De nombreuses recherches ont été menées pour minimiser la consommation électrique du processeur. Certains de ces travaux portent sur la modification de l'ordonnanceur de processus [5, 9]. Dans ces approches la technique la plus utilisée est l'adaptation dynamique de tension et de fréquence du processeur nommée «DVS» («dynamic voltage scaling») [6].

Certaines solutions reposent sur l'observation de la charge processeur et l'application d'algorithme de gestion de «DVS». Par exemple l'ordonnanceur de processus PAST [8] calcule sur des intervalles de temps de 10 à 50 milli-secondes l'utilisation passée du processeur. Si le temps d'inactivité du processeur est important alors l'ordonnanceur diminue la fréquence du processeur. Si ce temps d'inactivité est faible l'ordonnanceur augmente la fréquence. Cet algorithme n'est pas optimum s'il y a de grandes variations de charge du processeur à l'intérieur de l'intervalle de temps considéré.

L'approche Vertigo [3] répond en partie à ce problème. Cette solution consiste à ajuster automatiquement les performances du processeur sans participation de l'application. Il coexiste avec l'ordonnanceur natif. Vertigo utilise une hiérarchie d'algorithmes tous spécialisés dans différents types de situations. Pour les applications interactives, Vertigo utilise un algorithme spécifique pour diminuer l'impact du ralentissement du système vis à vis de l'utilisateur. Par exemple si le système détecte le lancement d'une application interactive la fréquence du processeur est augmentée. Cette approche est efficace avec des applications multimédias. Les algorithmes de gestion du «DVS» utilisés ne sont pas modifiables.

Pouwelse et al [7] déplacent l'algorithme de décision de la fréquence du processeur dans l'application multimédia. Ils proposent que l'application multimédia prévoit la quantité de traitement nécessaire au décodage de chaque trame. L'application indique au système la vitesse du processeur nécessaire pour son exécution. L'utilisation du gestionnaire est dépendante de l'exécution de l'application multimédia. Cette solution est uniquement adaptée sur un système d'exploitation mono-tâche.

Application multimédia coopérative

Pour illustrer l'inefficacité des gestionnaires de «DVS» nous étudions dans la seconde section de ce papier la consommation processeur d'une application multimédia. À partir de cette analyse nous proposons une solution de coopération à la problématique de décision entre l'ordonnanceur de processus et l'application multimédia. Notre solution met en œuvre une participation de la part de l'ap-

plication pour choisir une fréquence de fonctionnement du processeur. Notre solution s'adapte aisément avec d'autres algorithmes d'ordonnancement de processus et d'autres algorithmes de gestion de «DVS». La fréquence calculée par une application est prioritaire sur celle calculée par le système. Dans le cas où plusieurs applications choisissent une fréquence l'ordonnanceur choisit la plus importante. L'ordonnanceur utilise un algorithme de gestion de «DVS» par défaut qui est supplanté par ceux des applications à calcul de fréquence.

2 Application Multimédia et charge du processeur

Nous étudions la consommation de la ressource processeur par une application multimédia. Nous avons effectué nos tests sur un Dell Inspiron 510m équipé d'un processeur Pentium M 735. Ce processeur fonctionne avec des fréquences de 600 Mhz à 1,7 Ghz. Pour la lecture vidéo nous avons utilisé l'application Mplayer (`www.mplayerhq.hu`) exécutée sur un système Linux 2.4.24.

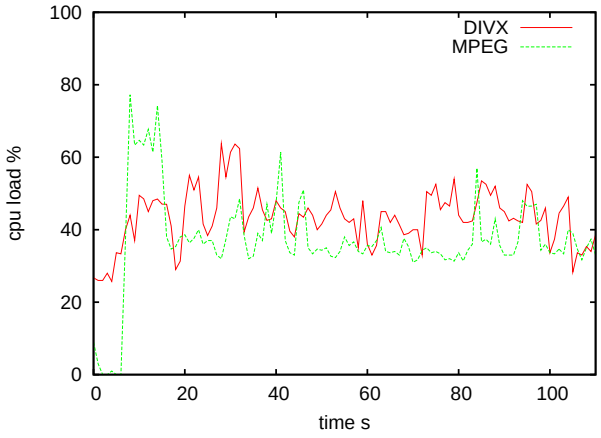

FIG. **1. Lecture vidéo et charge processeur**

La figure 1 représente la charge du processeur dans le temps lors de la lecture de deux vidéos par Mplayer. Il s'agit de la même vidéo, la bande annonce du film d'animation «Madagascar» dans deux formats d'encodage différents, divx 5 pour la vidéo `DIVX` et mpeg 2 pour la vidéo `MPEG`. Pour les deux fichiers le format est de 25 images de 1024 par 768 pixels par seconde, d'une durée de 110 secondes. Nous avons utilisé l'outil de monitoring mpstat [1] pour calculer la charge du processeur. Mplayer est exécuté dans un environnement sans gestionnaire de fréquence du processeur où la fréquence du processeur est de 1,7 Ghz. Mplayer ne supprime pas de trames à la lecture qui s'effectue avec un rendu optimal.

Un algorithme de gestion du «DVS» comme PAST utilisé sur des systèmes d'exploitations généralistes est inefficace lors de l'exécution d'une application telle que Mplayer dont la charge processeur varie énormément dans le temps et pour lequel les seuils de charge fréquemment utilisés sont 20% et 80%. Nous constatons que la charge processeur est comprise entre ces valeurs. Pourtant le système peut réaliser une économie significative de la consommation électrique. Par exemple durant les 20 premières secondes de la lecture du fichier `DIVX` la charge processeur est inférieur à 50 %. Le processeur utilisé consomme 7 W à 900 MHz contre 21 W à 1,7 Ghz, alors en appliquant une réduction de performance de 50% l'économie théorique d'énergie est de 66%.

3 Application coopérative et «DVS»

nous proposons une solution qui est de permettre à l'application Mplayer d'interagir avec le gestionnaire de «DVS». En utilisant au mieux la ressource processeur on économise de l'énergie tout en respectant les contraintes temporelles de l'application. Notre solution est mixte. Le système exploite par défaut l'algorithme PAST. Si une application est exécutée et qu'elle est instrumentée pour choisir une fréquence de fonctionnement alors l'algorithme PAST devient inactif.

Ordonnanceur

Nous effectuons notre évaluation sur un système Linux avec un noyau Bossa [2]. Bossa permet de développer et d'intégrer facilement et de façon sûr des ordonnanceurs de processus. La politique d'ordonnancement de processus initial Linux est modifiée pour intégrer l'algorithme PAST. Tout les n clockticks l'ordonnanceur applique l'heuristique suivante. (1) Si la charge processeur est supérieure à 80% la fréquence du processeur est augmentée. (2) Si la charge processeur est inférieure à 20% cette fréquence est diminuée. La valeur de la variable n est calculée en fonction de la durée d'un clocktick système. La durée entre n clockticks est comprise entre 10 et 50 ms suivant les spécifications de l'algorithme PAST. Pour permettre à une application de spécifier sa fréquence processeur de fonctionnement dans la politique d'ordonnancement Bossa Linux + PAST nous implémentons la fonction d'interface `set_freq`. Ce choix de fréquence est prioritaire par rapport à l'algorithme PAST. À la fin de l'exécution d'une application à calcul de fréquence l'ordonnaneur recherche la plus grande fréquence de fonctionnement dans la liste des processus. Si ce résultat égal à 0 le gestionnaire PAST est réactivé.

Application multimédia

L'étude de la charge processeur dans la seconde section illustre l'inefficacité d'un gestionnaire de «DVS» tel que PAST avec des applications multimédias. Il est donc plus intéressant dans le cas de l'exécution d'une application multimédia que celle ci calcule sa fréquence de fonctionnement. Ce calcul peut être effectué avant ou après le traitement de décodage du média. En utilisant une approche à posteriori notre solution fonctionne quel que soit le format de compression de la vidéo.

Dans le module A-V correction de la figure 2, Mplayer calcule le retard de traitement entre la vidéo et le son, le `AV_delay`. Mplayer utilise cette information pour estimer le temps disponible lors du traitement de la série d'images à venir. Dans le module sleep Mplayer utilise ces données pour décider combien de temps l'application peut attendre avant de continuer le traitement. La fonction utilisée pour provoquer cette attente est `usec_sleep` qui bloque le processus. Le processus n'est plus éligible par l'ordonnanceur de processus. Dans le module A-V correction la variable `AV_delay` est utilisé pour décider si la lecture de la vidéo s'effectue dans de bonne conditions de respect des contraintes temporelles. Si la valeur de cette variable est supérieur à 0.5 seconde la qualité du traitement est jugée médiocre. Dans ce cas Mplayer tente de corriger le traitement en supprimant l'affichage des prochaines trames.

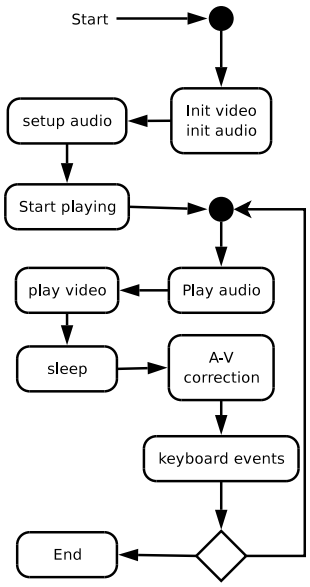

FIG. **2. Mplayer**

Nous modifions Mplayer pour qu'il coopère avec l'ordonnanceur de processus. Nous ajoutons un module `coop_dvs` avant l'exécution du module Sleep. Ce module décide en fonction de la valeur de `AV_delay` d'effectuer une demande de modification de fréquence du processeur. L'étude de la variable `AV_delay` dans notre environnement nous indique que la coopération entre application et système se réalise dans de bonne conditions pour des valeurs seuils de 0,2 et 0,25. Si la variable `AV_delay` est comprise entre ces deux valeurs la ressource processeur est suffisante pour effectuer le traitement du fichier vidéo. En deçà l'application demande au système de diminuer la fréquence et au delà d'augmenter la fréquence. Nous choisissons cet intervalle pour cette variable suite à des tests que nous avons effectués en mesurant les changements de fréquence et le rendu du traitement de la vidéo.

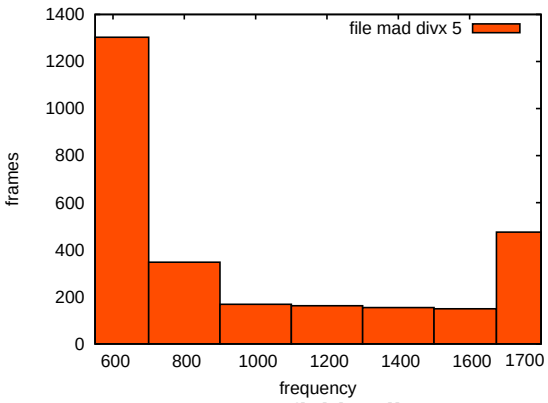

FIG. **3. DVS et fichier divx 5**

Résultats

Le diagramme de la figure 3 est le résultat de la lecture par Mplayer modifiée du fichier vidéo «Madagascar» au format divx 5 utilisé dans l'évaluation de la section 2. Les colonnes représentent le nombre d'images affiché à chaque fréquence, par exemple Mplayer affiche 1300 images sur les 2750 du fichier en utilisant le processeur à une fréquence de 600 Mhz.

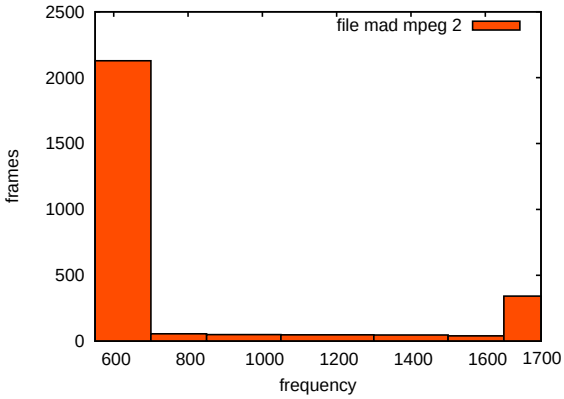

FIG. **4. DVS et fichier mpeg 2**

Le diagramme de la figure 4 est le résultat de la lecture par Mplayer modifiée du fichier vidéo «Madagascar» au format mpeg 2 utilisé dans l'évaluation de la section 2.

La consommation électrique d'un processeur notée P est proportionnelle à $V^2.f$ [4], V la tension en Volt et f la fréquence du processeur en Hz, P est en UE unité arbitraire. La consommation électrique pour la lecture du fichier vidéo est proportionnelle à $\sum_{i=1}^{n} {V_i}^2.f_i$ avec i le numéro de l'image affichée compris entre 1 et n.

L'économie d'énergie théorique de la lecture du fichier `DIVX` en utilisant le Mplayer modifié est de 57% par rapport à une exécution au maximum de fréquence du processeur. Ce gain théorique est de 70% pour la lecture du fichier `MPEG`.

4 Conclusion

La gestion coopérative de la ressource énergétique entre une application et le système d'exploitation est efficace. Cette approche est adaptée au cas des applications multimédias et de leurs contraintes temporelles. Les économies théoriques réalisées au niveau du processeur sont significative. Dans le cas des hiérarchies d'ordonnanceurs l'intégration de plusieurs gestionnaires de «DVS» reste à étudier.

Dans la suite de nos travaux nous désirons valider nos approches en les confrontant à des mesures réelles de la consommation électrique du système. Nous développons actuellement une plate-forme, la Bossa-Box (`www.emn.fr/x-info/bossa/bossabox/`) pour tester et valider ces approches de gestions énergétiques avec des applications nécessitant beaucoup de ressources. Par exemple nous effectuons la lecture différée d'un enregistrement vidéo en cours d'acquisition.

Références

[1] R. Andresen. Monitoring linux with native tools. In *30th Annual International Conference of The Computer Measurement Group, Inc.*, Las Vegas, Nevada USA, Dec. 2004.

[2] L. P. Barreto and G. Muller. Bossa : A DSL framework for application-specifi c scheduling policies. In IEEE, editor, *Eighth IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII). May 20–23, 2001, Schloss Elmau, Germany*, pages 161–161, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2001. IEEE Computer Society Press.

[3] K. Flautner and T. N. Mudge. Vertigo : Automatic performance-setting for linux. In *Proceedings of the 5th ACM Symposium on Operating System Design and Implementation (OSDI-02)*, Operating Systems Review, pages 105–116, New York, Dec. 9–11 2002. ACM Press.

[4] D. Genossar and N. Shamir. Intel pentium m processor power estimation, budgeting, optimization and validation. *Intel Technology Journal*, 7(2) :44–49, May 2003.

[5] K. Govil, E. Chan, and H. Wassermann. Comparing algorithms for dynamic speed-setting of a low-power CPU. In *Proceedings of the first Conference on Mobile Computing and Networking MOBICOM'95*, Mar. 1995. also as technical report TR-95-017, ICSI Berkeley, Apr. 1995.

[6] J. Lorch and A. Smith. Software strategies for portable computer energy management. *IEEE Personal Communications Magazine*, 5(3) :60–73, June 1998.

[7] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *MobiCom '01 : Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 251–259, New York, NY, USA, 2001. ACM Press.

[8] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 13–23, 1994.

[9] A. Weissel and F. Bellosa. Process cruise control : Event-driven clock scaling for dynamic power management. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems CASES'02*, Oct. 2002.