الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
كلية العلوم الدقيقة والتطبيقية
Faculté des sciences exactes et appliquées

Compte Rendu

Insertion d'une tache d'endormissement dans un taskset pour la réduction de la consommation d'énergie dans les système temps réels

Auteur : YEMMI Smail

Encadreur : Dr. OLEJNIK Richard
Encadreur : Dr BENYAMINA Abou-ElHassen
Co encadré par : Dr ZAHAF Houssam-Eddine

Table des matières

Table des figures

Liste des tableaux

Introduction Generale

Première partie

Partie Theorique

Chapitre 1

Introduction Au systeme temps réel

1.1 Introduction

1.2 Taxonomie sur les systèmes temps réel

Différents niveaux de criticité

Les systèmes temps réel dits critiques (ou dur) correspondent ont des systèmes pour lesquelles il est intolérable qu'une échéance soit manquée au risque de causer des conséquences graves, telles que des blessures ou des pertes humaines. Les centrales nucléaires ou le guidage de missiles représentent de tels systèmes à haute criticité. Dans le domaine de l'informatique embarqué, l'automobile et l'aéronautique regorgent de systèmes critiques à l'image des équipements déclencheurs d'airbags ou des logiciels de contrôle de vol de satellite. Il est crucial que les résultats soient disponibles au moment voulu et un résultat obtenu trop tard est inutilisable, à l'instar d'un système anti-missile qui recevrait la position d'un objet volant avec du retard.

Les systèmes temps réels mou sont des systèmes où on tolère les retards et ne requièrent pas un déterminisme temporel aussi fort que les systèmes temps réels dur. Par exemple, un logiciel de diffusion de flux vidéo produit un certain nombre d'images dans un intervalle de temps régulier. Le fait de manquer une ou plusieurs échéances ne provoque pas l'arrêt du système multimédia. La qualité de la vidéo est dégradée mais le service peut continuer de fonctionner sans risque. Donc les systèmes temps réels mou offre le meilleur service possible (notion de best effort) et les retards dans l'obtention des résultats ne sont pas dramatiques.

A la frontière entre les systèmes temps réel dur et mou, les systèmes temps réel ferme tolèrent une certaine proportion d'échéances manquées. Ils ne considèrent que les résultats obtenus à temps et sont liés à la notion de qualité de service (QoS).

1.3 Ordonnancement monoprocesseur

Un algorithme d'ordonnancement est chargé de répartir les tâches sur un ou plusieurs processeurs : il décide quelle tâche sera exécutée sur tel processeur et pour combien de temps.

Definition

Nous définissons dans un premier temps les termes habituels concernant les systèmes temps réel.

Hors-ligne en-ligne. Un algorithme d'ordonnancement hors-ligne prend la totalité de ses décisions d'ordonnancement avant l'exécution du système. Au contraire, un ordonnancement en-ligne prend les décisions d'ordonnancement lors de l'exécution

Priorités. Les algorithmes d'ordonnancement temps réel peuvent être classés suivant leur utilisation des priorités pour choisir quelle tâche doit être ordonnancée.

Préemptif / non préemptif. Un algorithme d'ordonnancement préemptif est un algorithme d'ordonnancement qui peut arrêter l'exécution d'une tâche, i.e. la préempter, à tout moment lors de l'exécution. Au contraire, un algorithme d'ordonnancement non préemptif ne permet aucune préemption, un travail en cours d'exécution ne peut être arrêté.

Ordonnançabilité / Faisabilité. Un système de tâches est dit ordonnançable si un ordonnancement existe permettant de satisfaire toutes les contraintes temps réel. Un système de tâches est dit faisable s'il existe un algorithme d'ordonnancement permettant d'ordonnancer ce système de tâches sans aucune violation d'échéances.

Optimalité. Un algorithme d'ordonnancement est dit optimal s'il peut ordonnancer tous les ensembles de tâches ordonnançables par d'autres algorithmes d'ordonnancement existants.

1.3.1 Algorithme d'ordonnancement à priorité fixe

Rate Monotonic [1]

Rate Monotonic est un algorithme à priorité fixe introduit par Liu et Layland dans [1]. Cet algorithme affecte des priorités aux tâches inversement proportionnel à leur période : plus leur période est petite, plus la tâche est prioritaire. Un exemple de système de tâche ordonnancée par Rate Monotonic est donné table 1.1. La figure 1.5 est une représentation graphique de l'ordonnancement correspondant.

Théorème 3 [1]. Rate Monotonic est optimal pour l'ordonnancement de systèmes de tâches synchrones, indépendantes et à échéance sur requête en présence de préemption.

Théorème 4 [1] (Condition Suffisante). Un système temps réel composé de n tâches est ordonnançable par Rate Monotonic si :

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1) \tag{1.1}$$

Deadline Monotonic

Deadline Monotonic est un algorithme à priorité fixe introduit par Leung et Whitehead dans Cet algorithme est proche de celui de Rate Monotonic, à la différence que les priorités sont maintenant affectées en fonction de l'échéance relative de chaque tâche au lieu de leur période. Cet algorithme est optimal dans le cadre des algorithmes à priorité fixe pour des systèmes de tâches synchrones à échéance contrainte lorsque la préemption est autorisée. Monotonic et Deadline Monotonic se confondent.
Condition suffisante d'ordonnançabilité La condition suffisante d'ordonnançabilité est inspirée de la condition suffisante d'ordonnançabilité de Liu et Layland (cf. théorème 4) :

Théorème 6. Un système temps réel composé de n tâches est ordonnançable par Deadline Monotonic si la condition suivante est vérifiée :

$$U = \sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1) \tag{1.2}$$

Condition necessaire et suffisante d'ordonnançabilité Joseph et al ont proposé un test d'ordonnancabilité basé sur le pire temps de reponse R_i. Le pire temps de réponse est le moment ou la tache i de priorité p terminera son exécution quand les taches les plus prioritaire sont actifs avec elle en même temps.
soit Γ = $\tau_1, \tau_2, ...\tau_n$ un ensemble de n taches. Γ est ordonnancable sous deadline monotonic ssi :

$$\forall \tau_i \in \Gamma / R_i \leq D_i \tag{1.3}$$

$$R_i = \begin{cases} R_i^0 = C_i \\ R_i^{(k+1)} = C_i + \sum_{j \in pr(i)} \left\lceil \frac{R_i^{(k)}}{T_j} \right\rceil * C_j \end{cases} \tag{1.4}$$

1.4 Ordonnancement Multiprocesseur

1.5 Conclusion

Chapitre 2

Etat de l'art

2.1 Introduction

2.2 Les modèles de consommation d'énerie DVFS et DPM

2.3 Les états C-states du processeur

2.4 L'endormissement de processeur (Online VS Offline)

2.5 Le Modèle d'endormissement de Dsouza

2.6 Conclusion

Deuxième partie

Contributions et Implementation

Chapitre 3

Endormissement de tache sous priorité Fixe

3.1 Le cas monoprocesseur

3.2 Le cas multiprocesseur

Chapitre 4

Endormissement de tache sous priorité dynamique

4.1 Le cas monoprocesseur

4.2 Le cas multiprocesseur

Chapitre 5

Expérimentations

5.1 La génération des taches

5.2 La simulation

5.3 Discussions

Conclusion Generale

Bibliographie

[1] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM, 20(1) :46–61, 1973.