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
جامعة وهران 1 أحمد بن بلة
Université d'Oran 1

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

Chapitre 1

1.1 Introduction

Dans un système, un processeur exécute un nombre bien défini de tâches ce qui consomme de l'énergie et les ordonnance de tel façon qu'elles respecte leurs échéances. Notre but est d'assurer ces deux contraintes ; moins consommer l'énergie tout en gardant l'odonnançabilité de l'ensemble de taches en appliquant des algorithmes appropriés qui nous aident à conclurent notre travail.

Dans ce chapitre nous détaillons différents points : nous commençons par la terminologie d'une tâche avec quelques notations, ainsi que les modes de consommation d'énergie, nous définissons ensuite ce qu'est une période harmonique et ce qu'est un ordonnancement par un Deadline Monotonic (DM).Nous présentons par la suite deux algorithmes de génération d'ensemble de tache et d'insertion d'une tâche d'endormissement, et nous terminons par une expérimentation.

1.2 Notations et terminologie

Soit un ensemble de taches périodique $\{\tau_1, \tau_2, ...\tau_n\}$ comprenant n taches périodiques. Chacune avec un pire temps d'exécution C_i, une période T_i et une échéance $D_i \leq T_i$, l'ensemble de taches est ordonnées tel que $T_1 \leq T_2 \leq \ldots \leq T_n$.

On considère que toutes les taches ont un temps d'arrivé initial $O_i = 0\ i = 1..n$ et nous adoptons un ordonnancement préemptif à priorité fixe par tache appelé deadline monotonic. Le taux d'utilisation d'une tache τ_i est donné par $U_i = C_i/T_i$ et le taux d'utilisation total U_{tot} est donnée par $U_{tot} = \sum_{i=1}^{n} U_i$

1.3 Consommation d'énergie

La plupart des processeurs modernes sont doté de mode de fonctionnement, chaque mode consomme une quantité d'énergie E. Le processeur a également besoin de plus ou moins de temps pour entrer ou sortir d'un mode TS.
Par exemple un processeur a :

Un mode actif dans lequel le processeur consomme une grande une quantité d'énergie E_{active} mais il peut exécuté des taches en attente d'être traiter.

Un mode repos (sieste) où il consomme moins d'énergie E_{idle} que le mode actif. Mais, aucun traitement ne peut se faire et un peu de temps ST_{idle} doit être dépassé pour entrer ou sortir de ce mode.

Un mode sommeil profond où il consomme le moins d'énergie E_{sleep}, mais aucun traitement ne peut se faire et une quantité considérable de temps doit être passé pour entrer ou sortir de ce mode.

Nous présenterons une nouvelle technique efficace qui maximise le pourcentage de temps passé par un processeur en mode sommeil profond sans violer les contrainte temporelles de l'ensemble de taches. Dans ce fait cette technique utilise chaque unité de temps ou le processeur n'est pas en mode actif pour être passé en mode sommeil profond. Ainsi, le mode inactif peut être potentiellement éliminé avec une économie matérielle et une amélioration de la consommation d'énergie.

1.4 Période harmonique

Soit $\Gamma = \{\tau_1, \tau_2, ...\tau_n\}$ un ensemble de n taches. Γ est harmonique si toute tache $\tau_i \in \Gamma - \{\tau 1\}$ à une période $T_i = k_i T_1$ avec $k_i \in \mathbb{N}^*$

pour une efficacité d'ordonnancement on défini une période harmonique T_H par :

$$T_H = \begin{cases} T_1/2 & \text{si } \exists\, \tau_j \in \Gamma - \{\tau 1\}\ tel\ que\ T_j < 2T_1 \\ T_1 & \text{sinon} \end{cases}$$

1.5 Ordonnancement par Deadline Monotonic

Deadline monotonic (DM) est un algorithme d'ordonnancement à priorité fixe par tache, les priorités sont accordées au taches selon leurs écheances , la plus grande priorité est accordée à la tache de plus petites écheance.

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 $\Gamma = \tau_1, \tau_2, ...\tau_n$ un ensemble de n taches. Γ est ordonnancable sous deadline monotonic ssi $\forall \tau_i \in \Gamma / R_i \leq D_i$

R_i est donné par :

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

τ	C_i	D_i	T_i	R_i
1	5	9	10	9
2	4	7	15	4
3	6	15	30	29

TABLE 1.1 – Ensemble de taches

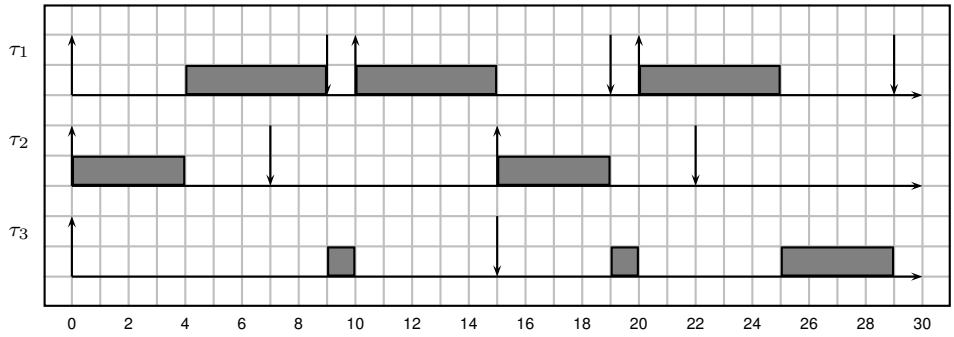

FIGURE 1.1 – Diagramme de Gantt

pour illustrer l'ordonnancement sous deadline monotonic posons l'exemple donnée dans le tableau 1.1 :

En utilisant l'équation , on obtient le résultat de la colonne R_i (voir le tableau 1.1). on remarque que le pire temps de réponse de la tache τ_3 est 29 et il est supérieur à son écheance qui est de 15. Donc, la tache n'est pas ordonnançable. Le diagramme de gantt (voir figure 1.1) de la simulation confirme le fait.

1.6 Algorithme de génération de tache

en 2005, Bini and Buttazzo [ref] ont crée un algorithme appelé UUnifast, il génère des taux d'utilisation de tache pour un ensemble de tache étant donnée un nombre de tache N et un taux d'utilisation total U_{tot}. En utilisant le principe de l'algorithme UUnifast on a abouti à un algorithme pour générer des taches .

Données : Taux Utilisation U_{tot}, Nombre de Taches N
Résultat : Ensemble de Taches Γ **Ordonnançable**
tant que Γ *non Ordonnaçable* **faire**
  $\Gamma \longleftarrow \emptyset$;
  $Sum \longleftarrow U_{tot}$;
  pour $i = 0$ **à** *N-1* **faire**
    $NextSum \longleftarrow Sum * Random()^{\frac{1}{(n-i)}}$;
    $UTask \longleftarrow Sum - NextSum$;
    $T \longleftarrow Random()$;
    $C \longleftarrow T * UTask$;
    $D \longleftarrow T * Random(0.75, 1)$;
    $tache \longleftarrow CreerTache(C, D, T)$;
    $\Gamma \longleftarrow \Gamma \cup tache$;
    $Sum \longleftarrow NextSum$;
  fin
  $T \longleftarrow Random()$;
  $C \longleftarrow T * Sum$;
  $D \longleftarrow T * Random(0.75, 1)$;
  $tache \longleftarrow CreerTache(C, D, T)$;
  $\Gamma \longleftarrow \Gamma \cup tache$;
fin

Algorithme 1 : Generation de taches

1.7 Algorithme d'insertion de tache

Posons une tâche d'endormissement $\tau_{sleep} = [C_{sleep}, D_{sleep}, T_{sleep}]$ tel que

- $D_{sleep} = T_{sleep} = T_H (T_H definiprecedemment)$
- $C_{sleep} = max\{C\}$ tel que $\Gamma \cup T_{sleep}$ est ordonnançable par Deadline monotonic

nous proposons cette algorithme qui génère une tache d'endormissement T_{sleep} et pour une performance de temps nous bornons C_{sleep} de tel façon que :

$$C_{sleepMin} \leq C_{sleep} \leq C_{sleepMax}$$

- $C_{sleepMin}$ la somme des temps pour entrer et sortir du mode sommeil profond
- $C_{sleepMax}$ = $(1 - U_{tot})T_{sleep}$

Données : TaskSet Γ, Temps Minimum d'execution $C_{SleepMin}$, Temps d'execution Maximum $C_{SleepMax}$
Résultat : Tache τ_{Sleep}
début
  $C_{Sleep} \longleftarrow C_{SleepMax}$;
  $D_{Sleep} \longleftarrow T_H$;
  $T_{sleep} \longleftarrow T_H$;
  tant que $\Gamma \cup tache(C_{Sleep}, D_{Sleep}, T_H)$ *est non ordonnançable* ***et*** $C_{Sleep} \geq C_{SleepMin}$ **faire**
    $C_{Sleep} \longleftarrow C_{Sleep} - \Delta C$;
  fin
  $\tau_{Sleep} \longleftarrow CreerTache(C_{Sleep}, D_{Sleep}, T_{sleep})$
fin

Algorithme 2 : Insertion Taches Endormissement

1.8 Experimentation

Nous avons pris des groupe de taskset harmonique de diffèrent taux d'utilisation allant de 0.1 à 0.8 on y a insérer des taches d'endormissement selon l'algorithme présenter ci-dessus.
Les deux graphes dans les figures 1.2 et 1.3 représente le minimum, le maximum et la moyenne des taux d'utilisation après insertion du différent groupe. On remarque que après insertion des taches d'endormissements, plus le taux d'utilisation de départ est faible plus le taux d'utilisation après insertion est considérable et grand. Donc, on conclut que dans les tasksets de taux d'utilisation faible il y a une perte d'énergie inutile en mode sieste alors qu'on peut gagner de l'énergie en basculant vers le mode sommeil profond.

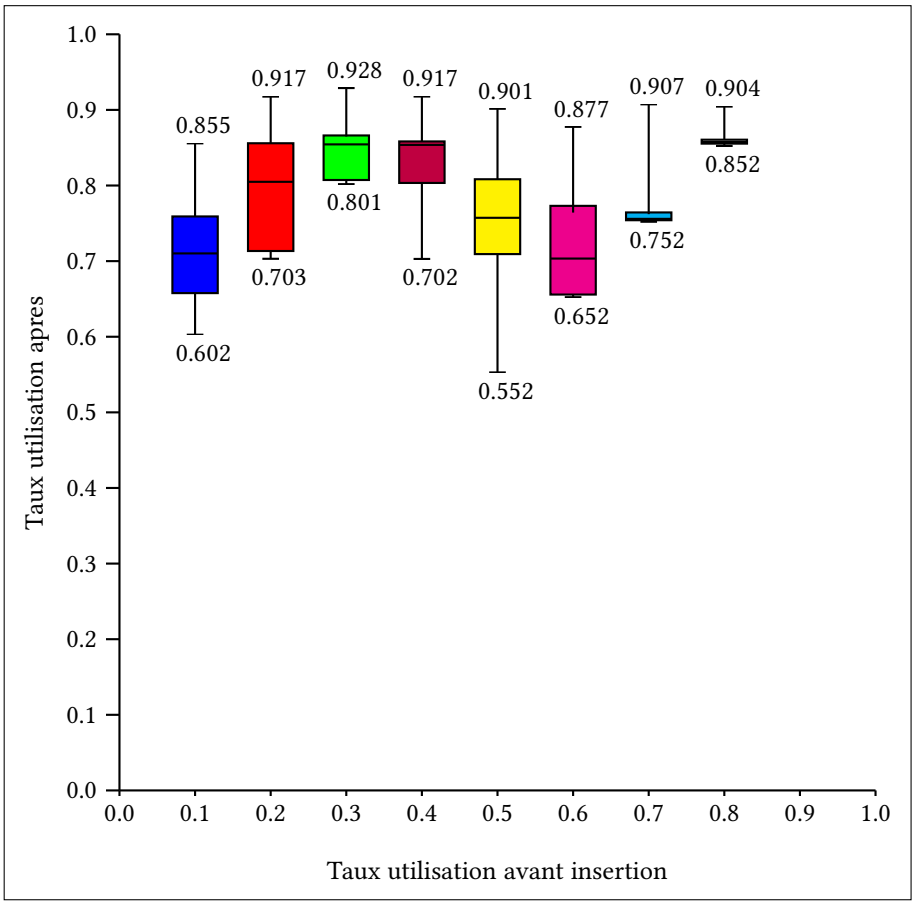

FIGURE 1.2 – Taux d'utilisation d'ensembles de taches harmoniques

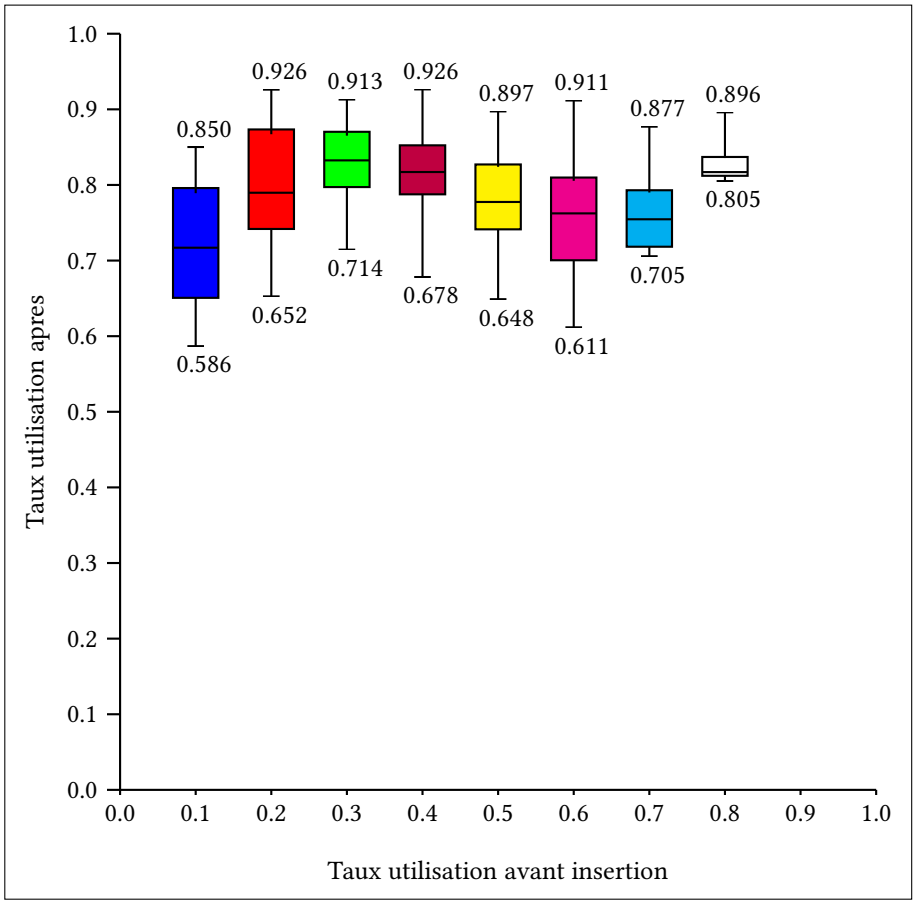

FIGURE 1.3 – Taux d'utilisation d'ensembles de taches non harmoniques