Sélection en ligne de la vitesse minimisant l'énergie dans les systèmes temps-réels

Bruno Gaujal (POLARIS), Alain Girault (SPADES) et **Stéphan Plassart**

Projet CASERM, Univ. Grenoble Alpes, Inria, CNRS, LIG, F-38000 Grenoble France.

Objectif

Sélectionner en-ligne la vitesse d'un processeur afin de minimiser l'énergie consommée par ce processeur exécutant un ensemble de tâches avec des contraintes temps-réel.

Etat de l'art

- Un seul processeur mono-coeur.
- Tâches indépendantes et sporadiques dont les jobs sont décrits selon le modèle ci-contre.

Objectif : **choisir la vitesse de processeur avec :**

1. Exécution de tous les jobs avant leurs échéances.
2. Minimisation de l'énergie totale consommée.

Solutions existantes :

- Cas **hors-ligne** : Yao et al. [1] ⇒ pas réaliste.
- Cas **en-ligne** : Bansal et al. [2] : ⇒ **Optimal Available (OA) :** A chaque instant t, on sélectionne la vitesse optimale en ne considérant que les jobs arrivés avant t.

Modèle

- Jobs (c_i, d_i) :
  - c_i : WCET ($\leq C$)
  - d_i : échéance relative ($\leq \Delta$)
- r_i : date d'arrivée du job i
- Vitesse du processeur : $v(t) \in \{0, v_1, ..., v_{max}\}$, v_{max} vitesse maximale.
- $j(v(t))$: puissance (active) dissipée par le processeur à vitesse v(t) en t. $j(.)$ convexe.
- $J = \int_0^T j(v(t))dt$: consommation d'énergie (active) du processeur de 0 à T (horizon de temps).

Problème de minimisation d'énergie

Trouver $v(t)$ $\forall t$ pour minimiser $\mathbb{E}\int_0^T j(v(t))dt$ sous contrainte que :

1. *Aucun job ne manque son échéance.*
2. *$v(t)$ ne dépend que de l'information disponible à t : $\mathcal{H}(t)$.*

Idée de départ

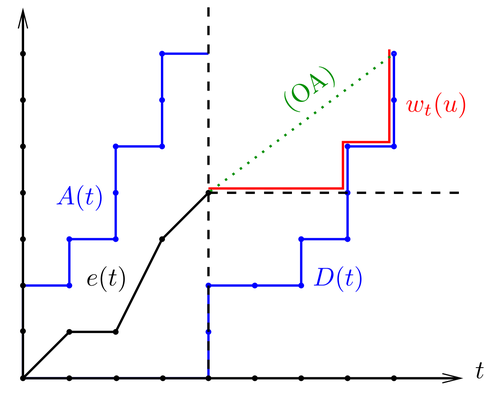
Utiliser des connaissances statistiques sur l'arrivée des jobs pour anticiper les prochaines arrivées de jobs dans le futur et faire mieux que (OA) pour la consommation d'énergie.

Information à t

$$\mathcal{H}(t) = \{(r_i, c_i, d_i) | r_i \leq t\} \cup \{v(u), u \leq t\}$$

Le choix de la vitesse ne dépend que d'informations passées ou actuelles.

Construction de la fonction de travail restant



Construction de la fonction de travail restant $w_t(\cdot)$ en t=4, pour les jobs :
J_0= (2,4), J_1 = (1,5), J_2= (2,6), J_3= (2,4), J_4= (0,6)
et les vitesses de processeur :
$v_0 = 1$, $v_1 = 0$,$v_2 = 2$, $v_3 = 1$.
- $A(t)$: quantité de travail arrivée avant t.
- $D(t)$: quantité de travail qui doit avoir été exécutée avant t.
- $e(t)$: quantité de travail déjà exécutée en t.

Processus à décision de Markov (PDM)

1. Nombre de jobs fini :

   Minimisation de l'espérance de la consommation d'énergie totale du processeur J^* de 0 à T, avec un état initial w_0 :

   $$J^*(w_0) = \min_v \left(\mathbb{E} \left(\sum_{t=0}^{T} j(v_t) \right) \right)$$

   ⇒ algorithme de programmation dynamique (DP)

2. Nombre de jobs infini :

   Minimisation de l'espérance de la consommation moyenne d'énergie par unité de temps, g :

   $$g^* = \min_v \mathbb{E} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} j(v_t) \right)$$

   ⇒ algorithme Itération des valeurs

⇒ Calcul, pour chaque état (w, t), de la vitesse optimale v^*.

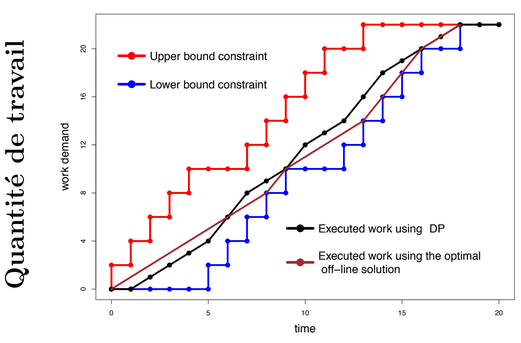
Mise en pratique

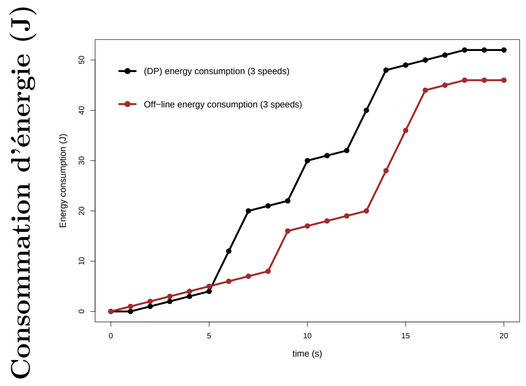
1. Détermination *hors-ligne* de la vitesse optimale du processeur pour chaque état possible du système (grâce au (PDM)).
2. *Mémorisation* de ces vitesses optimales dans une table.
3. Sélection *en-ligne*, en fonction de l'état courant du système, de la vitesse optimale en lisant la table.

Comparaison des résultats expérimentaux

(DP) vs hors-ligne jobs indépendants

- 60% du temps : jobs (2,5)

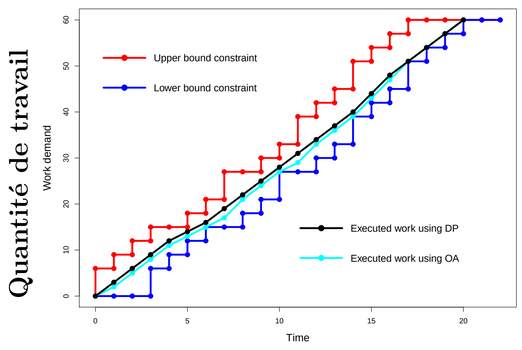


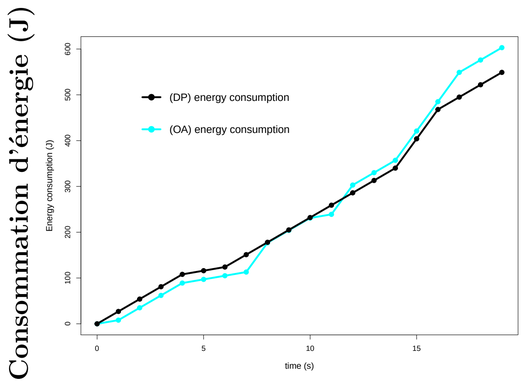


Surcoût = 12%

(DP) vs (OA) jobs indépendants

- 60% du temps : jobs (2,3)
- 20% du temps : jobs (4,3)

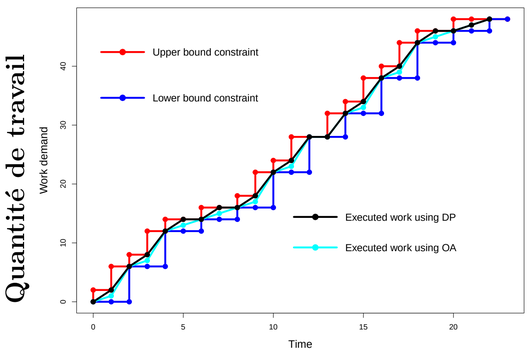


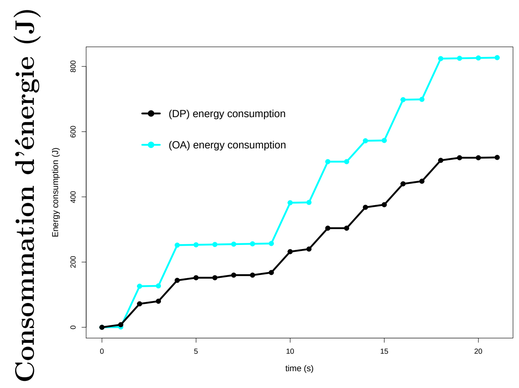


Gain = 10%

(DP) vs (OA) 2 tâches alternées

tâche 1 : 80% des temps pairs : jobs (2,2)
tâche 2 : 75% des temps impairs : jobs (1,4)





Gain = 60%

Quand les jobs sont plus prédictibles, le gain est plus important.

Extensions

1. Considérer d'autres politiques d'ordonnancement pour l'exécution des jobs : Quels résultats pour ces politiques ? Si on prend en compte le coût de préemption ?
2. Prendre en compte les coûts de changement de vitesse.
3. Proposer des algorithmes sous-optimaux plus rapides (en temps et en espace).
4. Utiliser des techniques d'apprentissage (comme le Q-learning) pour découvrir en-ligne les informations statistiques sur les jobs temps-réel.

[1] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proceedings of IEEE Annual Foundations of Computer Science*, 1995, pp. 374–382.

[2] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *Journal of the ACM*, vol. 54, no. 1, 2007.