Qualité de Service des Systèmes Temps réel Embarqués sous Contraintes d'Energie

A. Queudet

Journée thématique "Temps réel et énergie" - ACTRISS - GDR ASR - 17 janvier 2012

Plan de la présentation

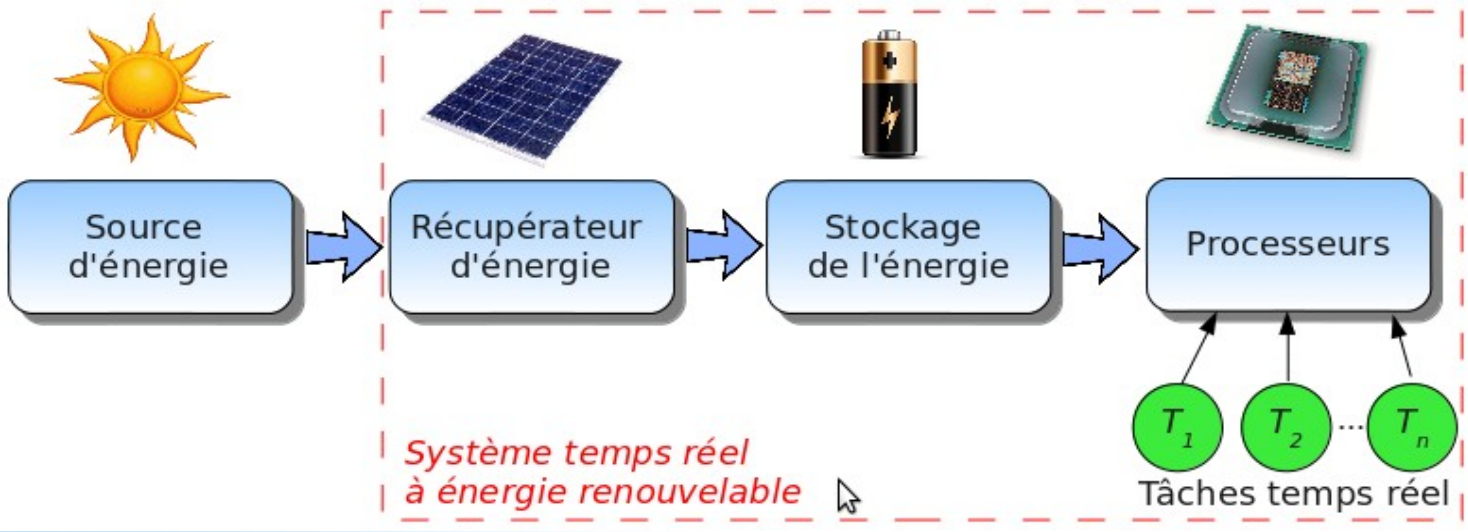
- Contexte et problématique
- Fondements théoriques
- Partitionnement “classique”
- Partitionnement sous contraintes de QoS
- Conclusions et perspectives

Plan de la présentation

- Contexte et problématique
- Fondements théoriques
- Partitionnement “classique”
- Partitionnement sous contraintes de QoS
- Conclusions et perspectives

Contexte et problématique (1)

- **Contexte**
  - *Systèmes temps réel multiprocesseur sous contraintes énergétiques*
    - Systèmes autonomes connectés à une *source d'énergie renouvelable*



Contexte et problématique (2)

- **Contexte**
  - *Applications temps réel sous contraintes de QoS*
    - une tâche temps réel peut, *occasionnellement* et *dans certaines limites spécifiées*, ne pas respecter ses échéances temporelles

Contexte et problématique (3)

- **Problématique**
  - *Partitionner et ordonnancer des tâches temps réel sous contraintes de QoS sur un système multiprocesseur contraint en énergie*
  - *Gérer les cas de famine énergétique*
    - Ensemble des traitements à effectuer > énergie disponible au sein du système

Plan de la présentation

- Contexte et problématique
- Fondements théoriques
- Partitionnement “classique”
- Partitionnement sous contraintes de QoS
- Conclusions et perspectives

Fondements théoriques (1)

- **Le modèle de QoS Skip-Over [KS95] :**

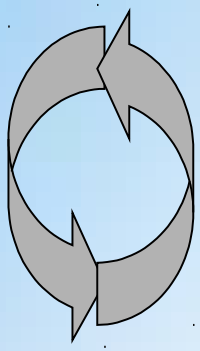
$$\tau_i = (C_i, T_i, D_i, s_i)$$

- des instances de tâche rouges

→ la tâche doit respecter son échéance

Paramètre de pertes (skip)

$2 \leq s_i \leq \infty$

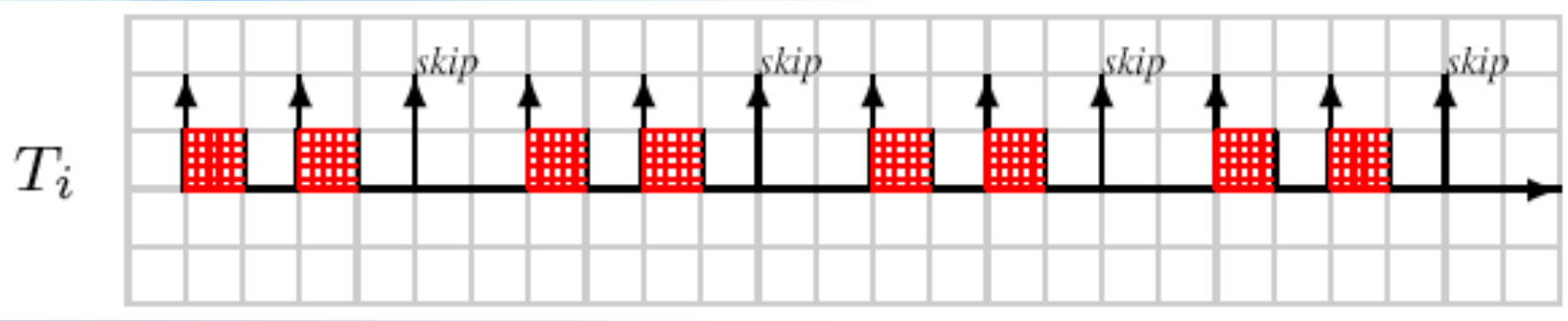


- des instances de tâche bleues

→ la tâche peut être abandonnée à tout moment

Fondements théoriques (2)

- **Algorithme RTO (Red Tasks Only) [KS95] :**
  - Toutes les instances de tâche sont rouges et elles sont ordonnancées selon EDF
  - Les instances bleues sont systématiquement rejetées
  - Exemple :

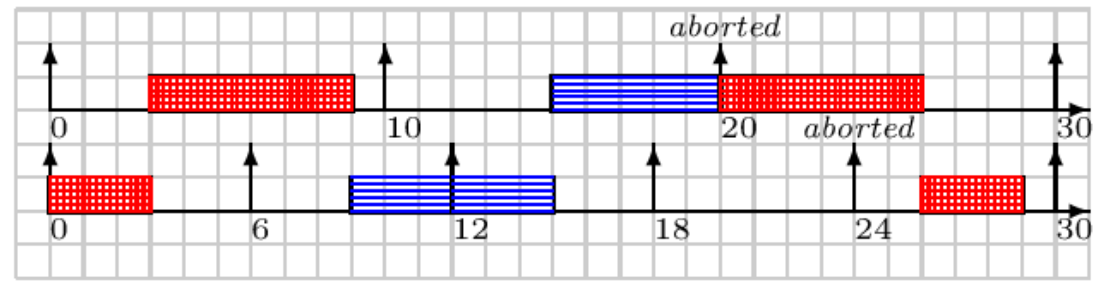


Fondements théoriques (3)

- **Algorithme BWP (Blue When Possible) [KS95] :**
  - Les instances de tâche peuvent être rouges ou bleues
  - Les instances de tâche rouges sont ordonnancées selon EDF
  - Les instances bleues sont élues lorsqu'il n'y a plus de rouges prêtes
  - Exemple :

T_1 (6, 10, 10, 2)

T_2 (3, 6, 6, 2)



Fondements théoriques (4)

- **Utilisation initiale du modèle Skip-Over :**
  - Systèmes temps réel monoprocesseur
  - Dégradation de la QoS en cas de surcharge du processeur
  - Garantie d'une QoS minimale
- **Utilisation dans notre cas :**
  - Systèmes temps réel multiprocesseur
  - Dégradation de la QoS en cas de famine énergétique
  - Garantie d'une QoS minimale

Plan de la présentation

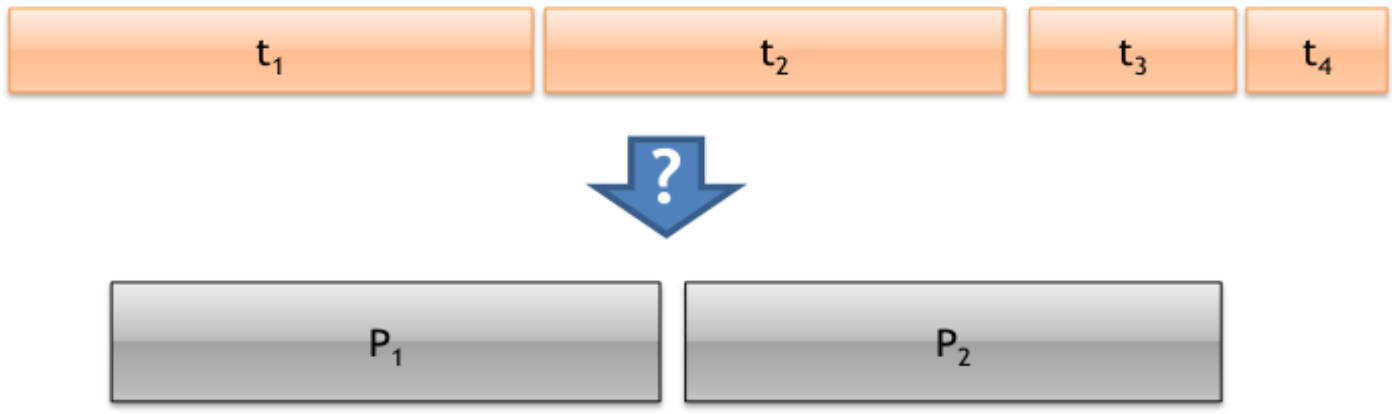
- Contexte et problématique
- Fondements théoriques
- Partitionnement “classique”
- Partitionnement sous contraintes de QoS
- Conclusions et perspectives

Partitionnement “classique” (1)

- Plateforme composée de *m* processeurs identiques $\pi = \{\pi_1, ..., \pi_m\}$
- Application composée de *n* tâches temps réel périodiques à échéances arbitraires : $\tau = \{\tau_1, ..., \tau_n\}$ avec $\tau_i(C_i, T_i, D_i)$
- Chaque tâche est caractérisée par :
  - Un facteur d'utilisation $u_i = C_i/T_i$
  - Une densité $\delta_i = C_i / (min(D_i, T_i)$

Partitionnement “classique” (2)

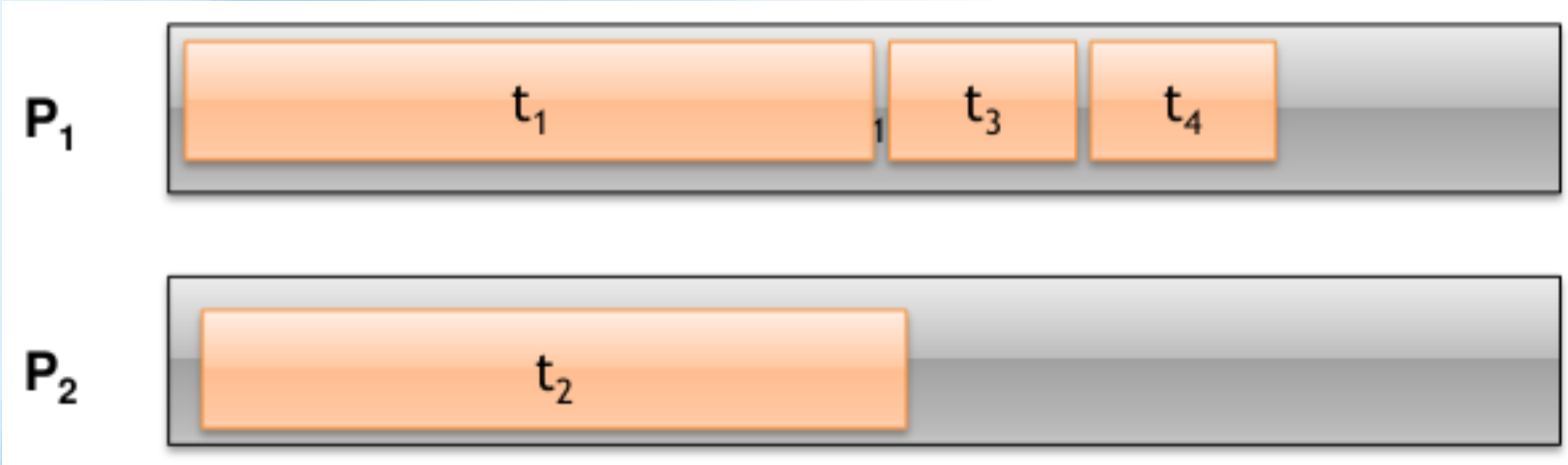
- **Principe :**
  - Tri de la liste des tâches
  - Affectation de chaque tâche dans l'ordre à un processeur selon un critère d'acceptation



  - Problème de Bin-Packing identifié comme *NP-difficile* [LW82]

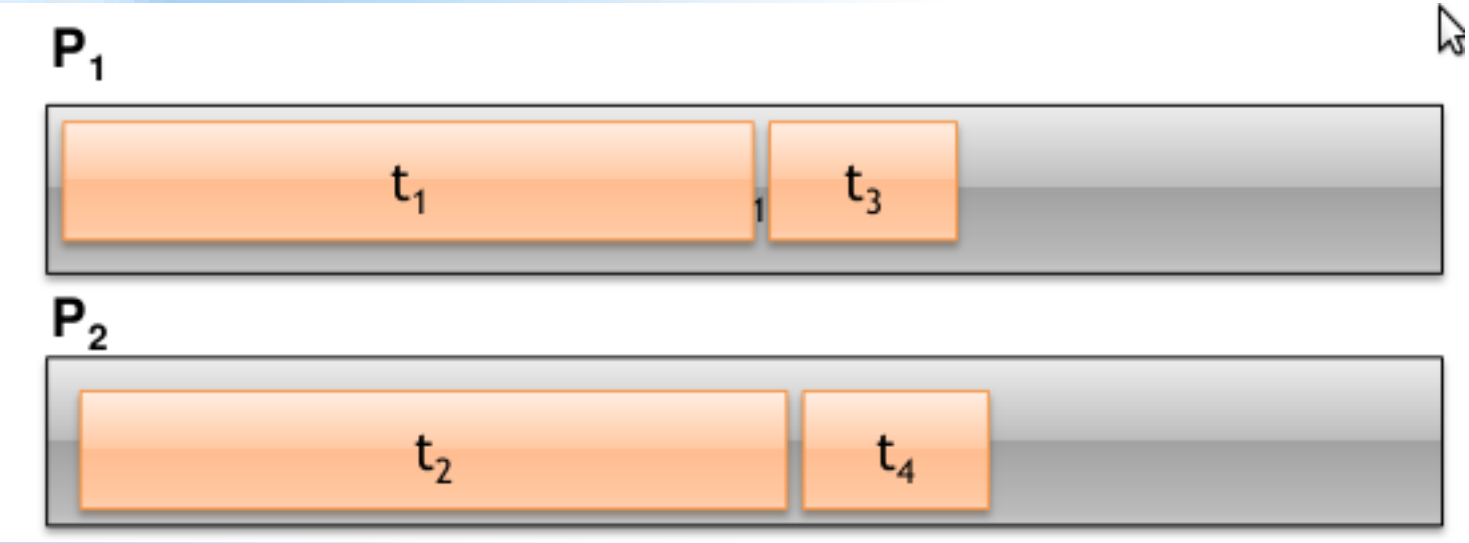
Partitionnement “classique” (3)

- **Heuristique First-Fit**
  - Une tâche est assignée au premier processeur susceptible de l'accepter



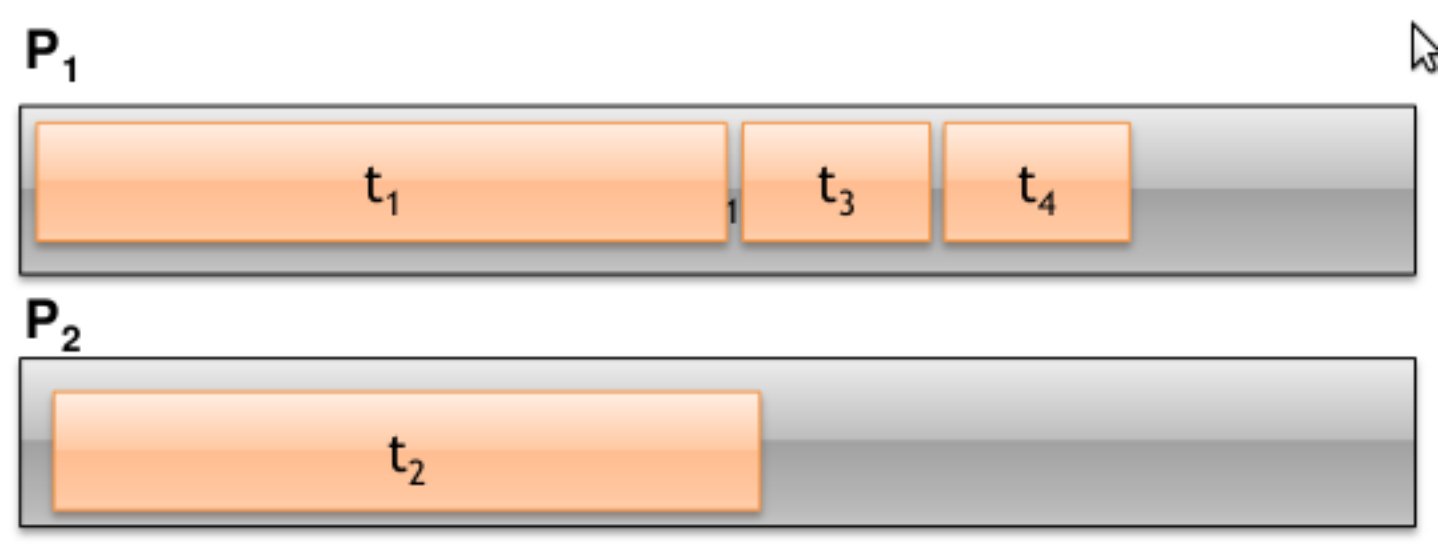
Partitionnement “classique” (4)

- **Heuristique Worst-Fit**
  - Une tâche est assignée au processeur de manière à maximiser la capacité processeur restante.



Partitionnement “classique” (5)

- **Heuristique Best-Fit**
  - Une tâche est assignée au processeur de manière à minimiser la capacité processeur restante



Partitionnement “classique” (6)

- **Critère d'acceptation d'une tâche sur un processeur**
  - *Test exact d'ordonnançabilité sous EDF :* $Load(\tau) \leq 1$
- Facteur de charge [BHR90] :

$$Load(\tau) = max\{U_\tau, sup_{L \in [D_{min}, P)} \frac{DBF(\tau, L)}{L}\}$$

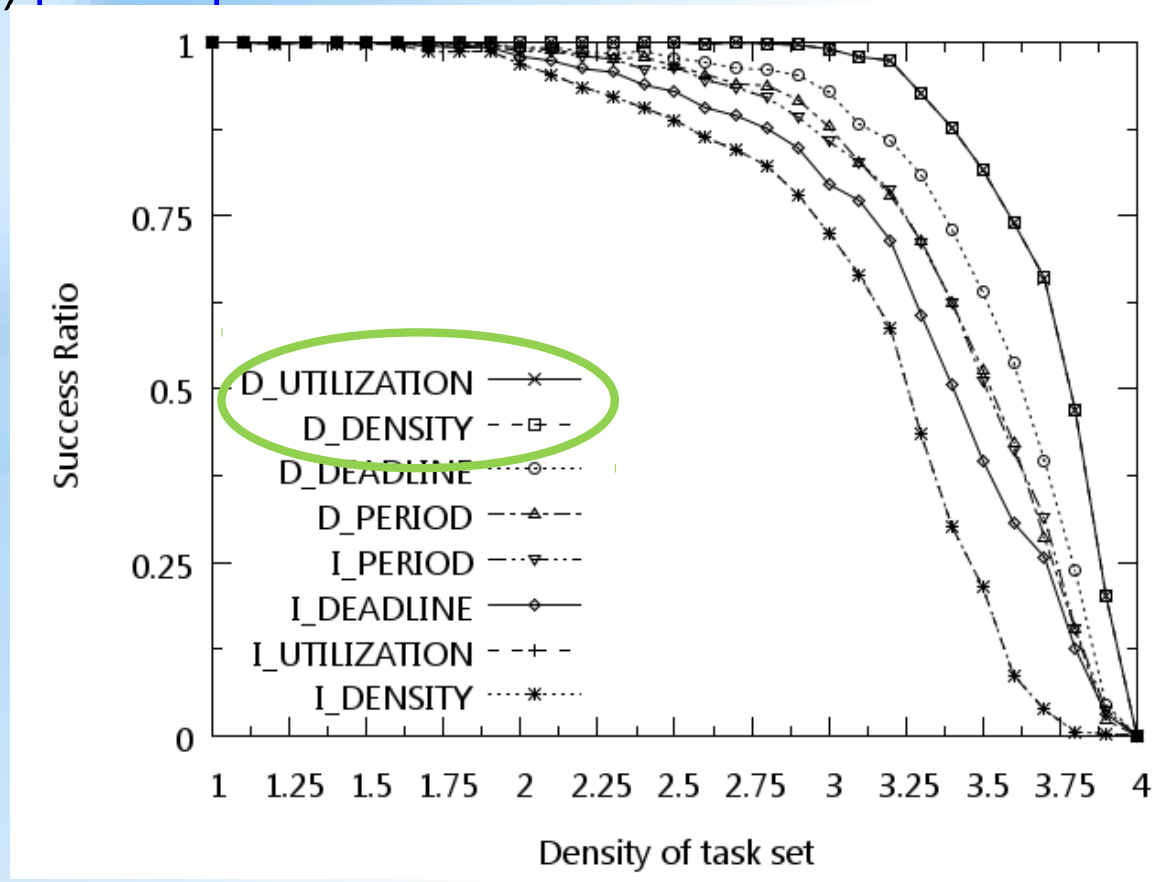
- $U_\tau = \sum_{i=1}^{n} \frac{C_i}{T_i}$
- $D_{min} = min\{D_1, ..., D_n\}$
- $P = lcm\{T_1, ..., T_n\}$
- $DBF(\tau, [0, L]) = \sum_{i=1}^{n} \left(1 + \lfloor \frac{L - D_i}{T_i} \rfloor\right) \times C_i$

Partitionnement “classique” (7)

- **Critères de performances :**
  - *Taux de succès* : nombre de configurations ordonnançables / nombre de configurations générées
  - *Nombre de processeurs utilisés*
  - *Valeur moyenne de la capacité processeur restante*

Partitionnement “classique” (8)

- Taux de succès en fonction du critère de tri des tâches (*m*=4, EDF) [LC+10] :



Plan de la présentation

- Contexte et problématique
- Fondements théoriques
- Partitionnement “classique”
- Partitionnement sous contraintes de QoS
- Conclusions et perspectives

Partitionnement sous contraintes de QoS (1)

- Plateforme composée de *m* processeurs identiques $\pi = \{\pi_1, \ldots, \pi_m\}$
- Application composée de *n* tâches temps réel périodiques sous contraintes de QoS : $\tau = \{\tau_1, \ldots, \tau_n\}$ avec $\tau_i(C_i, T_i, D_i, s_i)$
- Chaque tâche est caractérisée par :
  - Un facteur d'utilisation $u_i = C_i/T_i$
  - Une densité $\delta_i = C_i / (min(D_i, T_i)$
  - Un facteur d'utilisation équivalent $u_i = C_i/T_i * (s_i-1)/s_i$
  - Une densité équivalente $\delta_i = C_i / (min(D_i, T_i) * (s_i-1)/s_i$

Partitionnement sous contraintes de QoS (2)

- **Sélection d'un sous-ensemble d'heuristiques de partitionnement**
  - minimisation de la consommation énergétique globale du système
    - choix de *First-Fit* : minimisation du nombre de processeurs utilisés
  - flexibilité pour recharger le système en cas de famine énergétique
    - choix de *Worst-Fit* : maximisation de la capacité processeur restante
- **Adaptation de ces heuristiques de partitionnement à des tâches sous contraintes de QoS**
  - Nouveaux critères d'acceptation des tâches
    - Nouveaux tests d'ordonnançabilité intégrant les contraintes de QoS

Partitionnement sous contraintes de QoS (3)

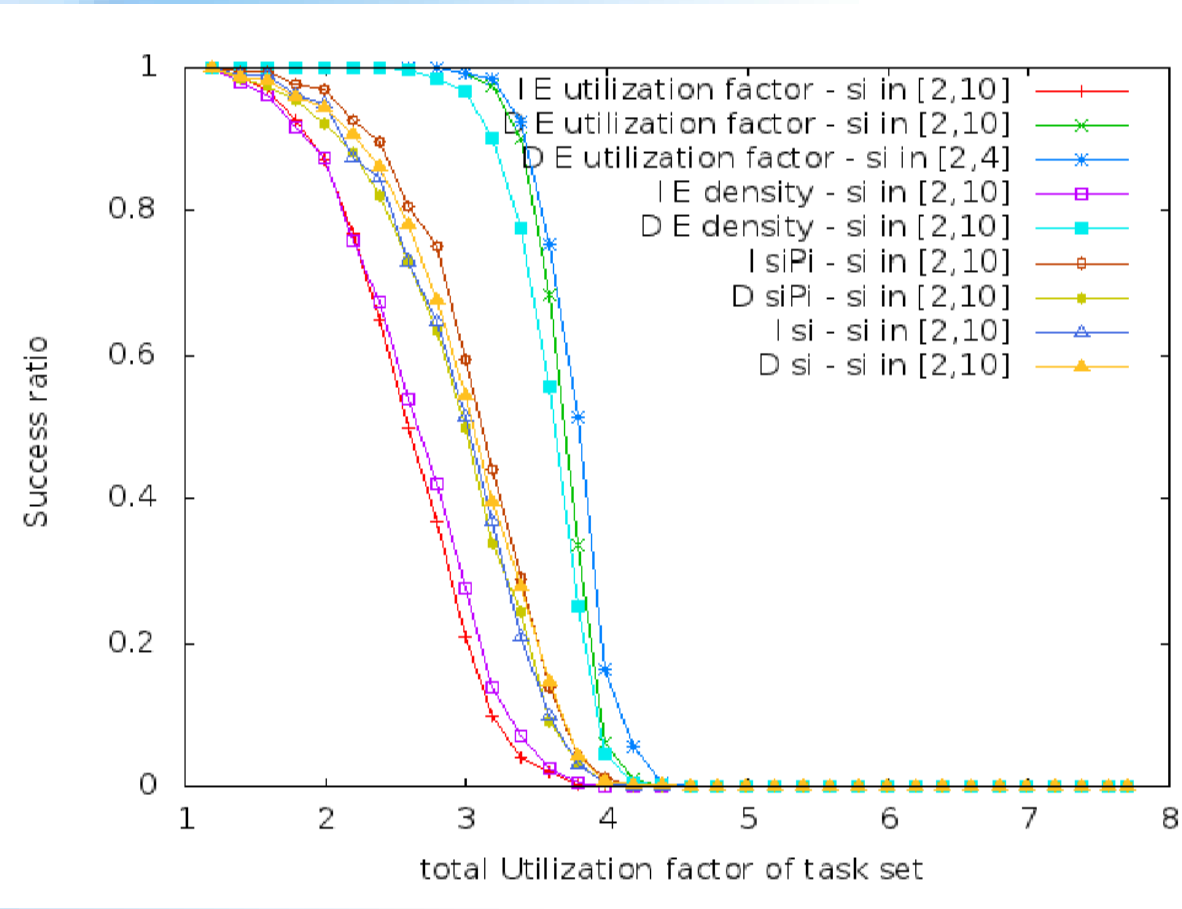
- **Critère d'acceptation d'une tâche sur un processeur**
  - *Test exact d'ordonnançabilité sous RTO/BWP :* $Load_{QoS}(\tau) \leq 1$
- Facteur de charge avec contraintes de QoS :

$$Load_{QoS}(\tau) = max\{U^*_\tau, sup_{L\in[D_{min},P)} \frac{DBF_{QoS}(\tau,L)}{L}\}$$

- $U^*_\tau = max_{L\geq 0}\{\frac{\sum_i D(i,[0,L])}{L}\}$ avec $D(i,[0,L]) = \lfloor \frac{L}{T_i} \rfloor - \lfloor \frac{L}{T_i s_i} \rfloor$
- $D_{min} = min\{D_1, ..., D_n\}$
- $P = lcm\{s_1 T_1, ..., s_n T_n\}$
- $DBF_{QoS}(\tau,[0,L]) = \sum_{i=1}^{n} \left(1 + \lfloor \frac{L-D_i}{P_i} \rfloor - \lfloor \frac{1}{s_i}\left(1 + \lfloor \frac{L-D_i}{T_i} \rfloor\right) \rfloor\right) \times C_i$

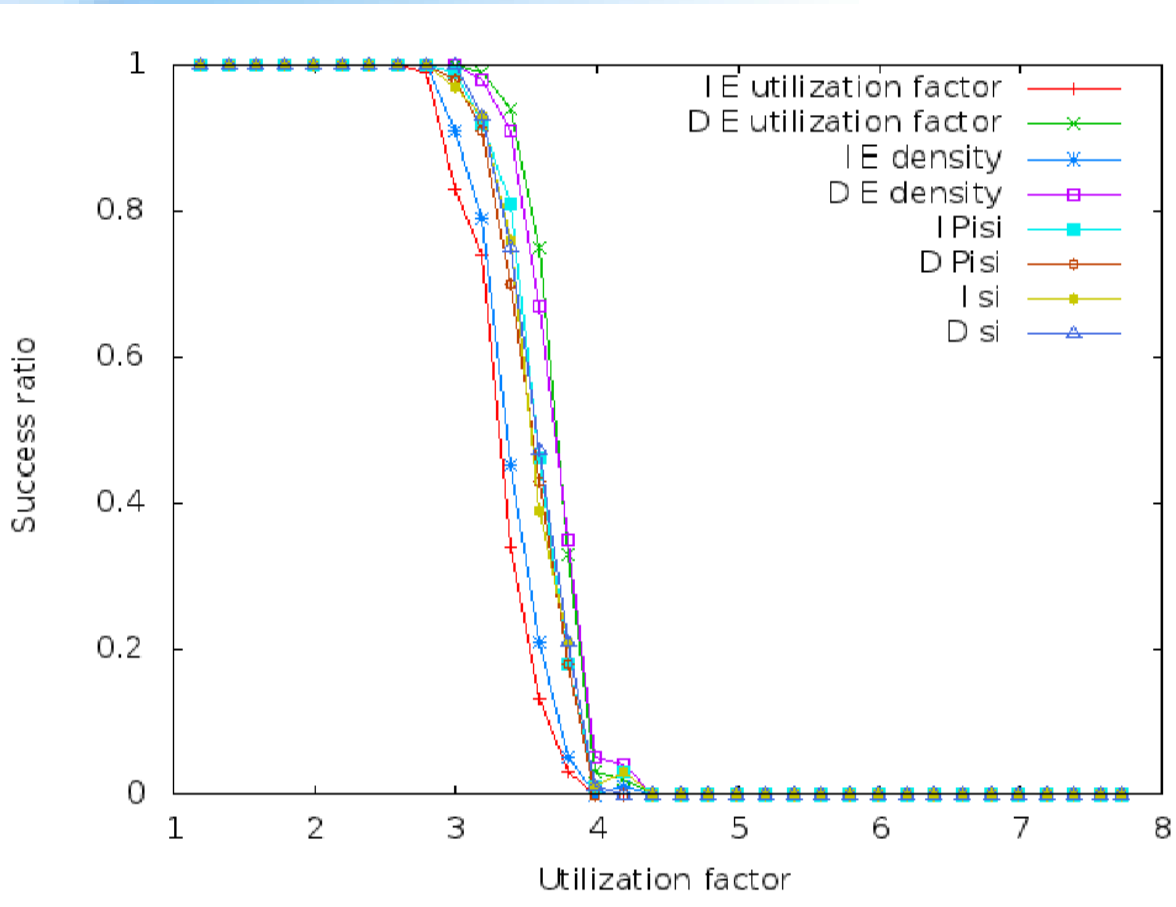
Partitionnement sous contraintes de QoS (4)

- Taux de succès en fonction du critère de tri des tâches (Worst-Fit) :



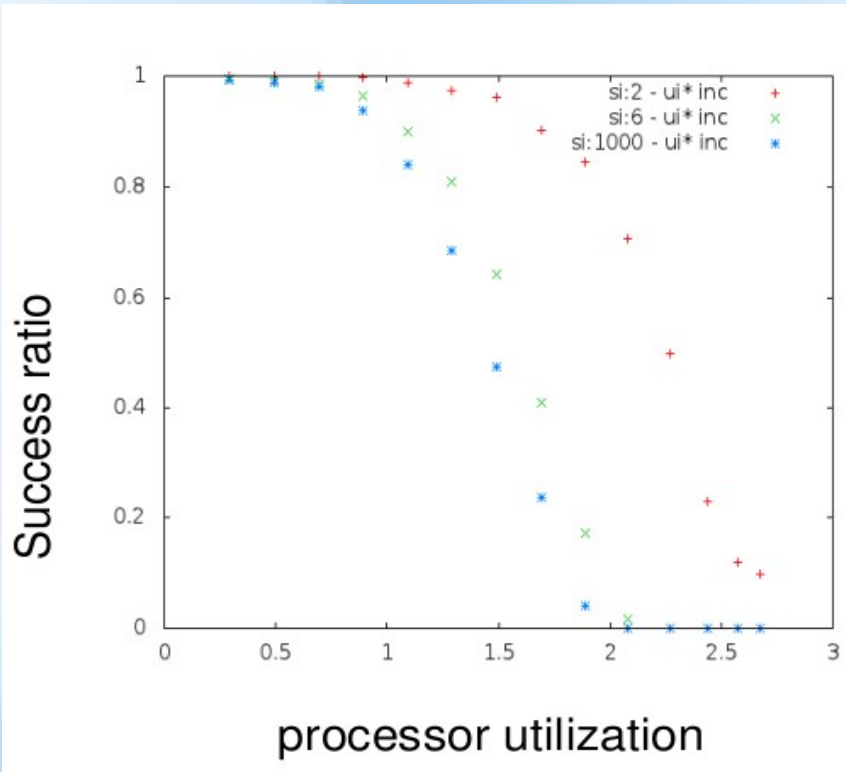
Partitionnement sous contraintes de QoS (5)

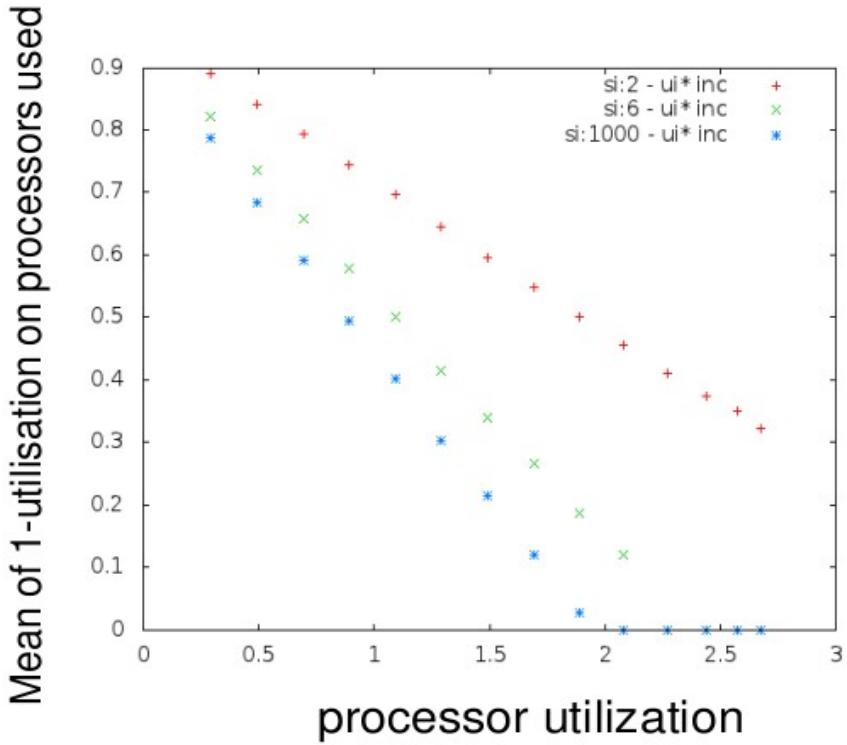
- Taux de succès en fonction du critère de tri des tâches (First-Fit) :



Partitionnement sous contraintes de QoS (6)

- Taux de succès et Valeur moyenne de la capacité processeur restante en fonction du paramètre de pertes des tâches :





Plan de la présentation

- Contexte et problématique
- Fondements théoriques
- Partitionnement “classique”
- Partitionnement sous contraintes de QoS
- Conclusions et perspectives

Conclusions et perspectives

- **Conclusions**
  - Tâches temps réel sous contraintes de QoS pour les systèmes multiprocesseur contraints en énergie
  - Sélection et adaptation de 2 heuristiques de partitionnement aux contraintes de QoS
  - Analyse d'ordonnançabilité pour le partitionnement sous contraintes de QoS
- **Perspectives**
  - Proposition d'autres heuristiques de partitionnement avec QoS
  - Adaptation des approches semi-partitionnées aux contraintes de QoS
    - Instances de tâche rouges migrantes
    - Instances de tâche bleues non-migrantes

Merci de votre attention...

Questions ?