Minimisation de Consommation d'Energie pour Tâches Temps Réels Parallèles sur Architectures Multicœurs Hétérogènes

Houssam Eddine ZAHAF

Directeurs de thèse:

Richard Olejnik

Abou El Hassan Benyamina

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Université de Lille

جامعة وهران 1
Université d'Oran 1 أحمد بن بلة

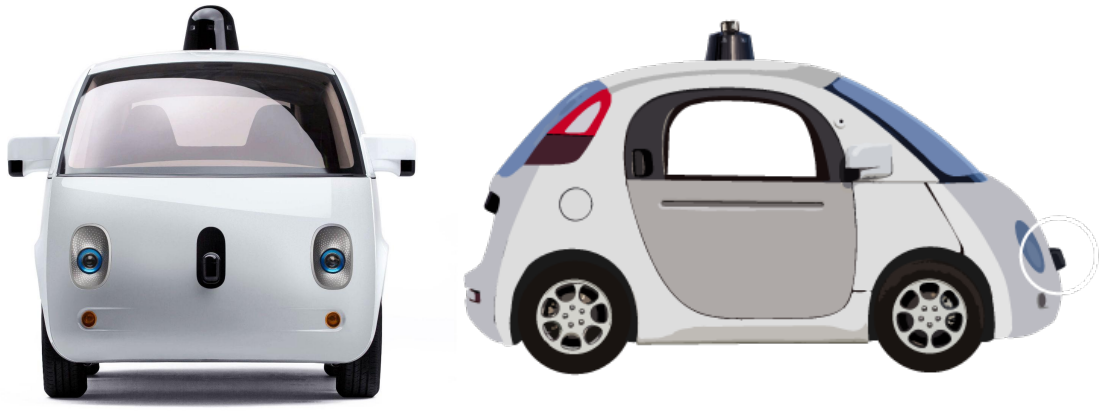
مختبر الهياكل المتوازية، المحمولة والحساب المكثف
LAPECI
Laboratoire des Architectures Parallèles, Embarquées et du Calcul Intensif.

02 Novembre 2016

Plan

Plan

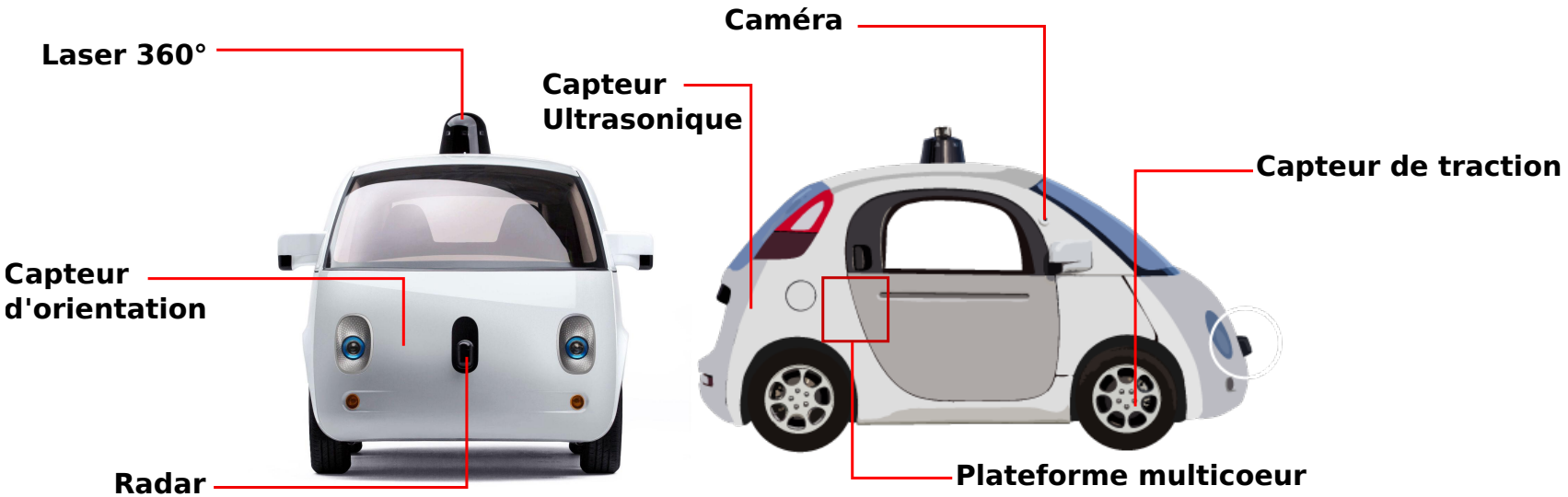
Introduction : Systèmes cibles



Le véhicule autonome

- Traiter une grande quantité de données.
- Exprimer un parallélisme intra-tâche.
- Contraintes :
  - Temps réel
  - Energie

Introduction : Systèmes cibles



Le véhicule autonome

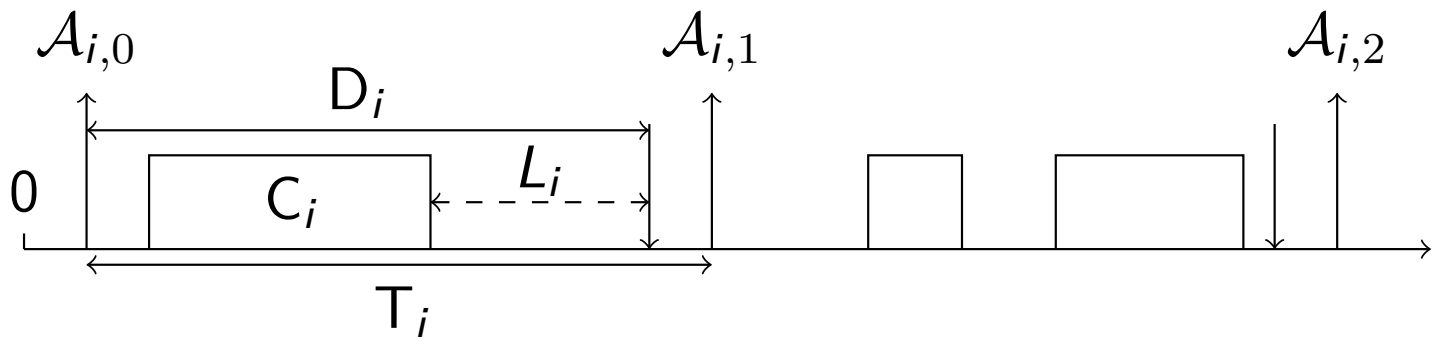
- Traiter une grande quantité de données.
- Exprimer un parallélisme intra-tâche.
- Contraintes :
  - Temps réel
  - Energie

Introduction : Systèmes temps réels

- Les résultats doivent être corrects **logiquement** et délivrés à **temps**.
  - Délivrés à temps $\neq$ le plus rapide
- Les tâches temps réels sont récurrentes

Introduction : Systèmes temps réels

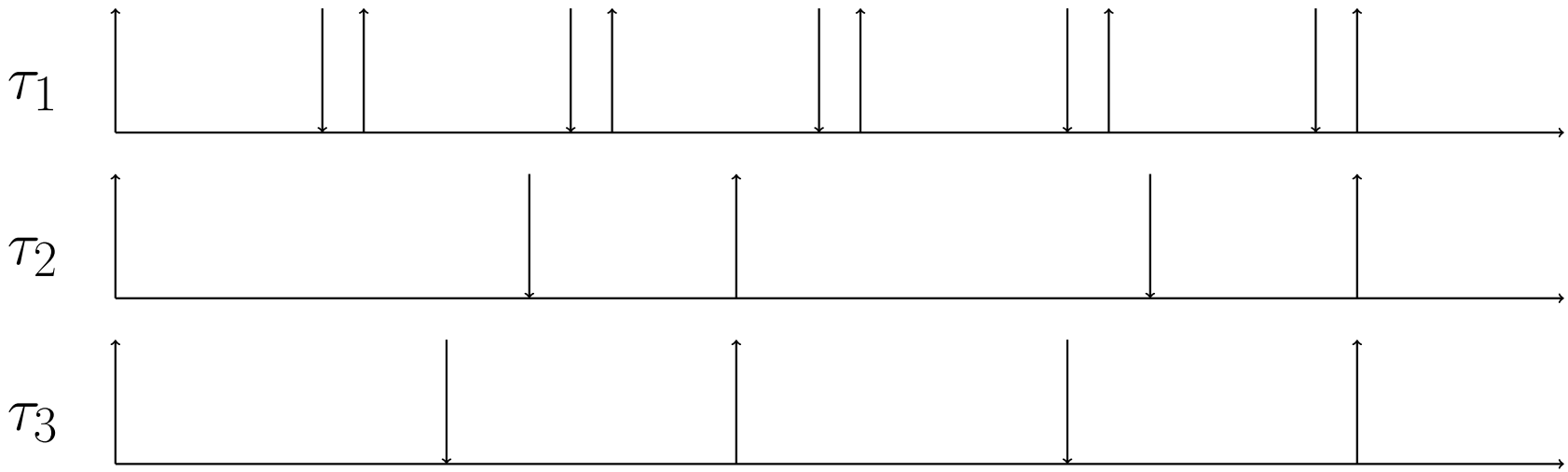
- Les résultats doivent être corrects **logiquement** et délivrés à **temps**.
  - Délivrés à temps $\neq$ le plus rapide
- Les tâches temps réels sont récurrentes



Le modèle de Liu and Layland

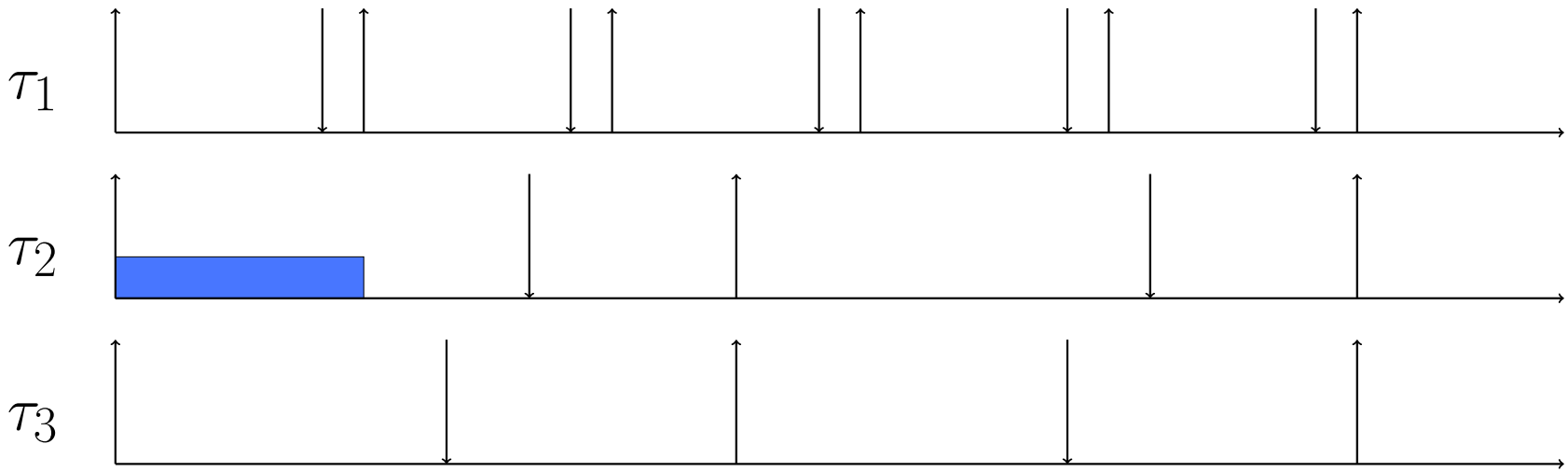
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



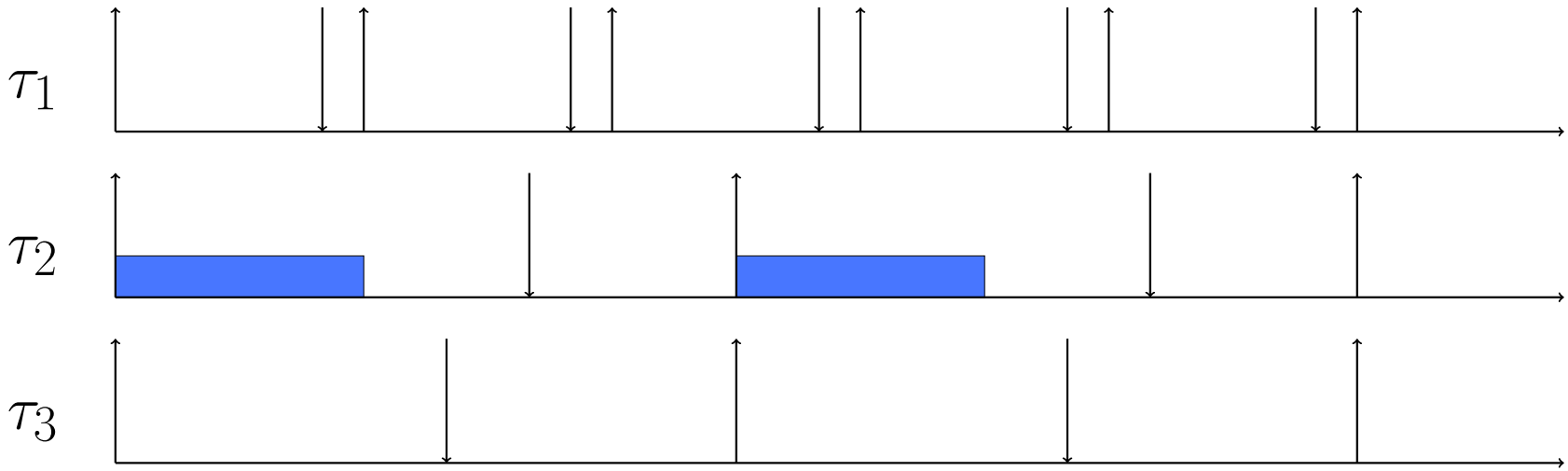
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



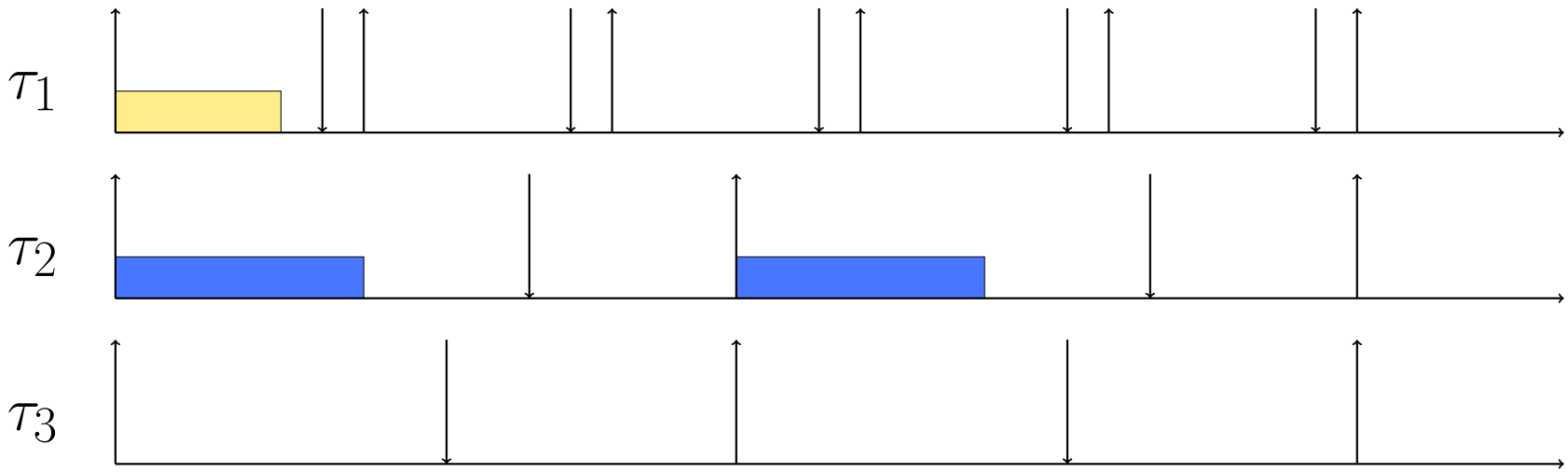
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



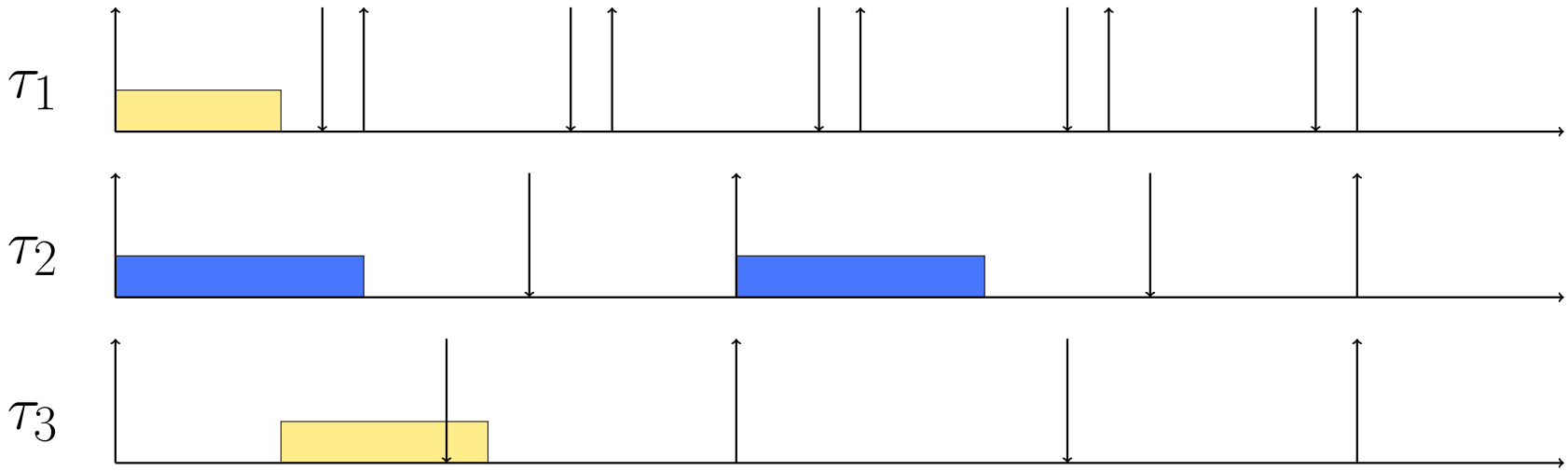
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



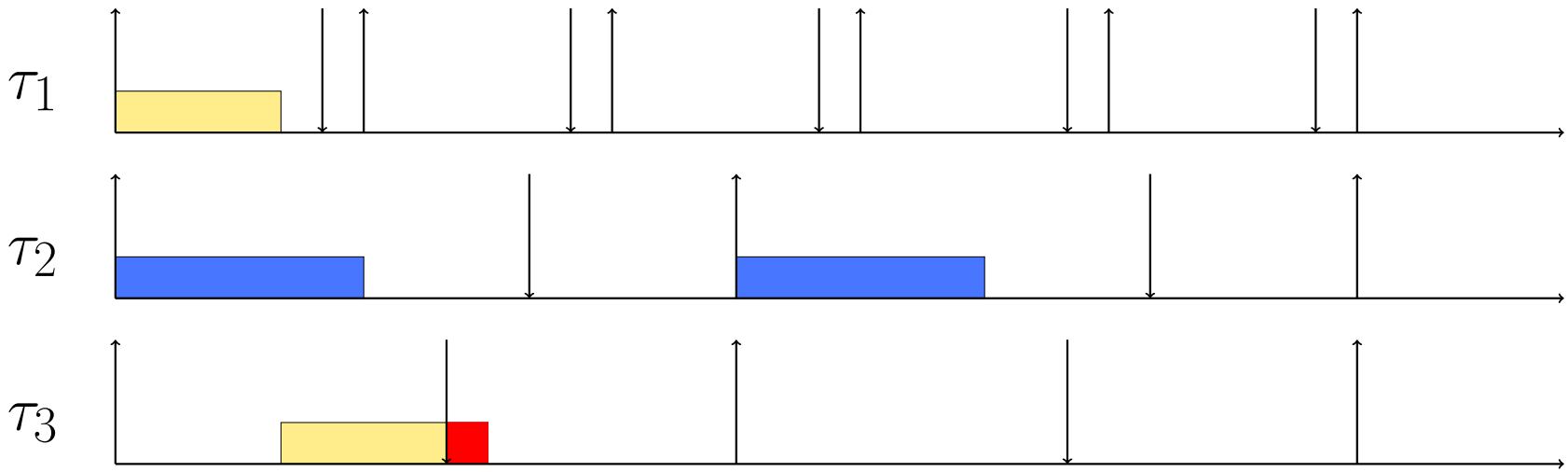
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



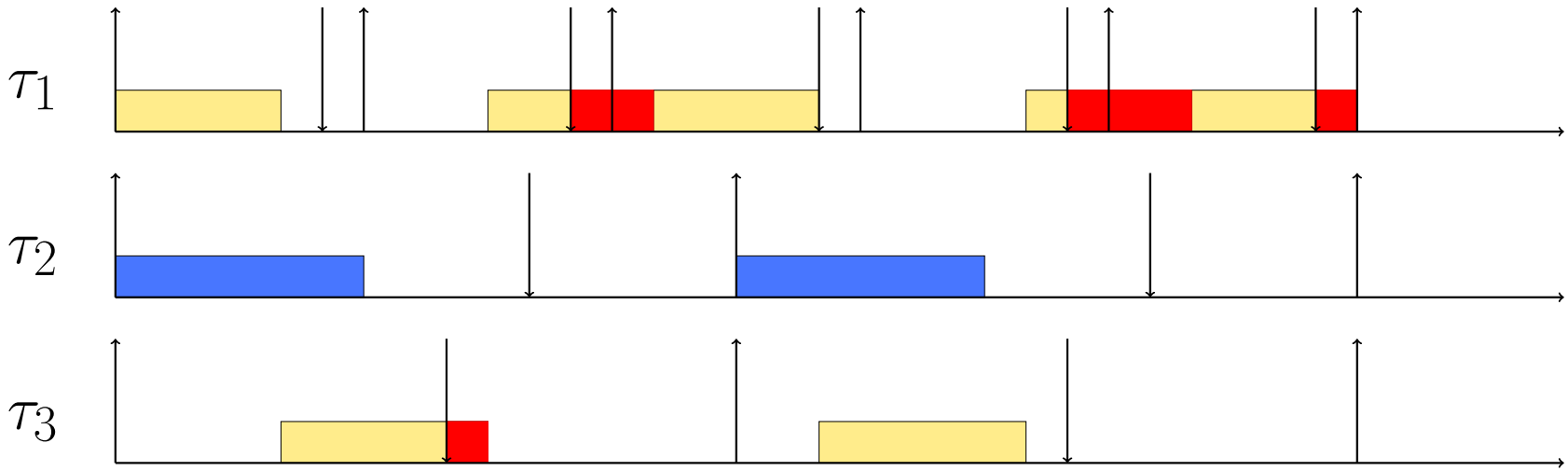
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



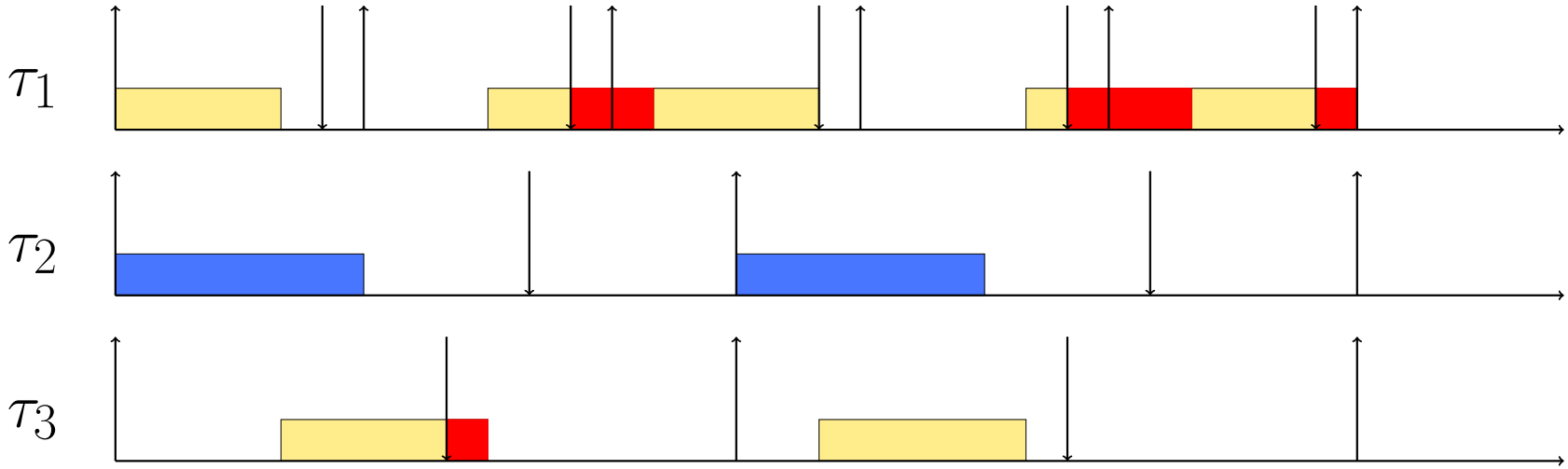
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu



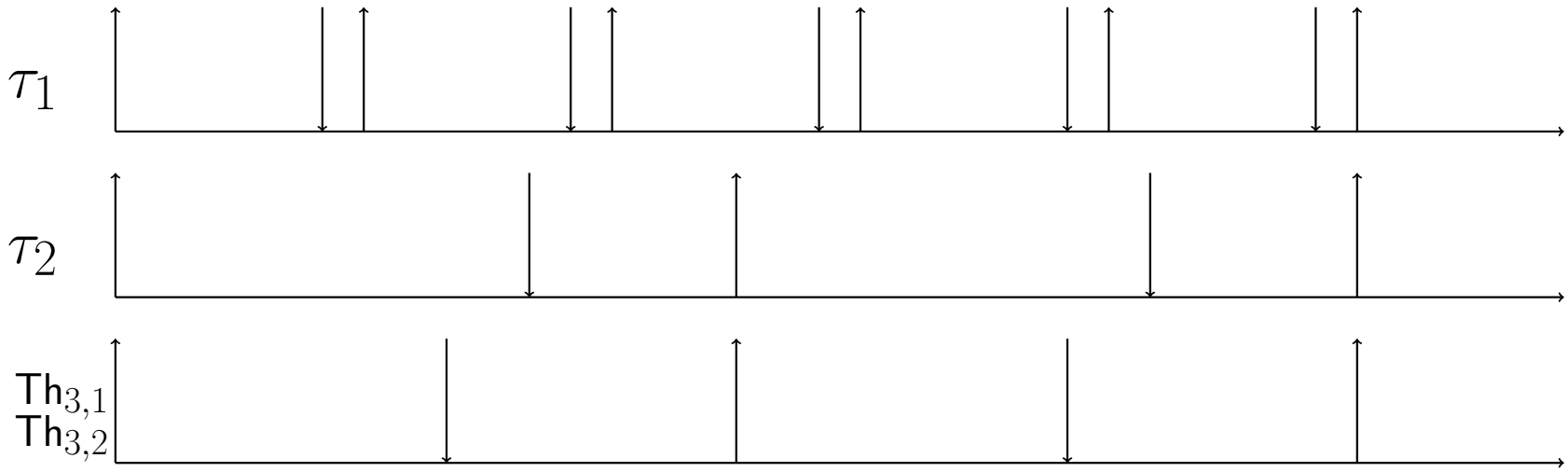
Ordonnancement temps réel EDF-Partitionné

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



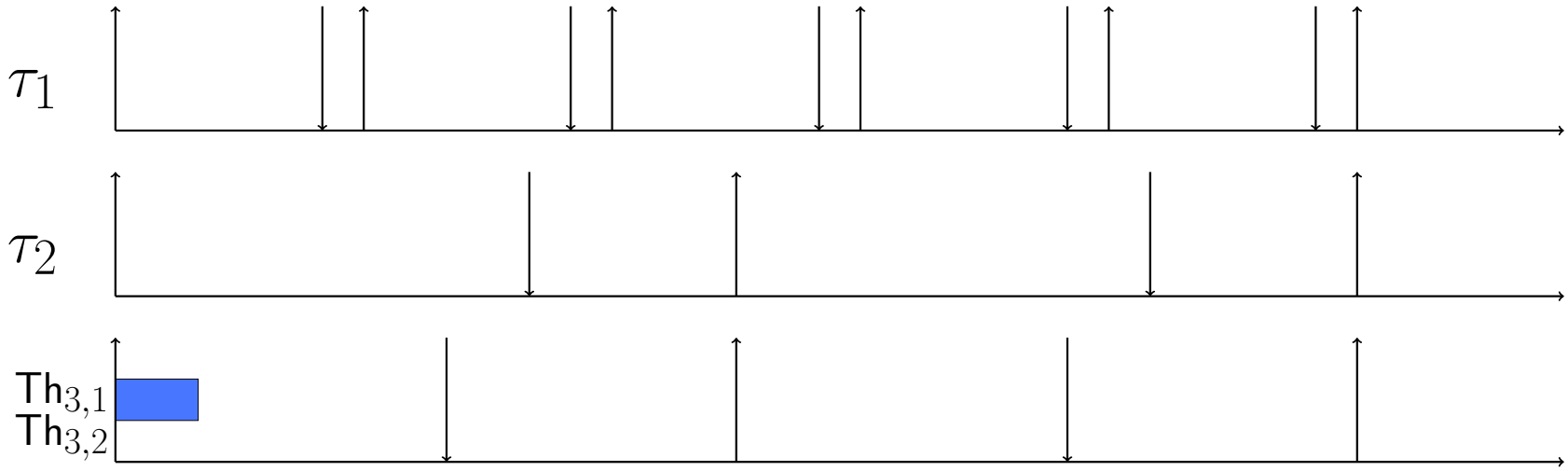
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



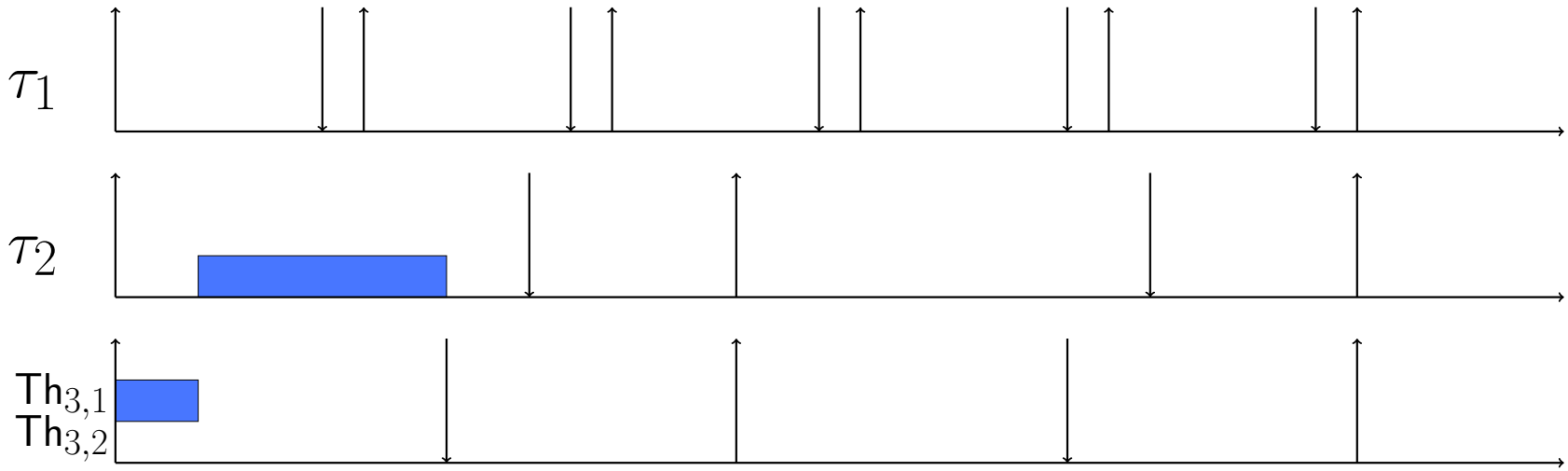
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



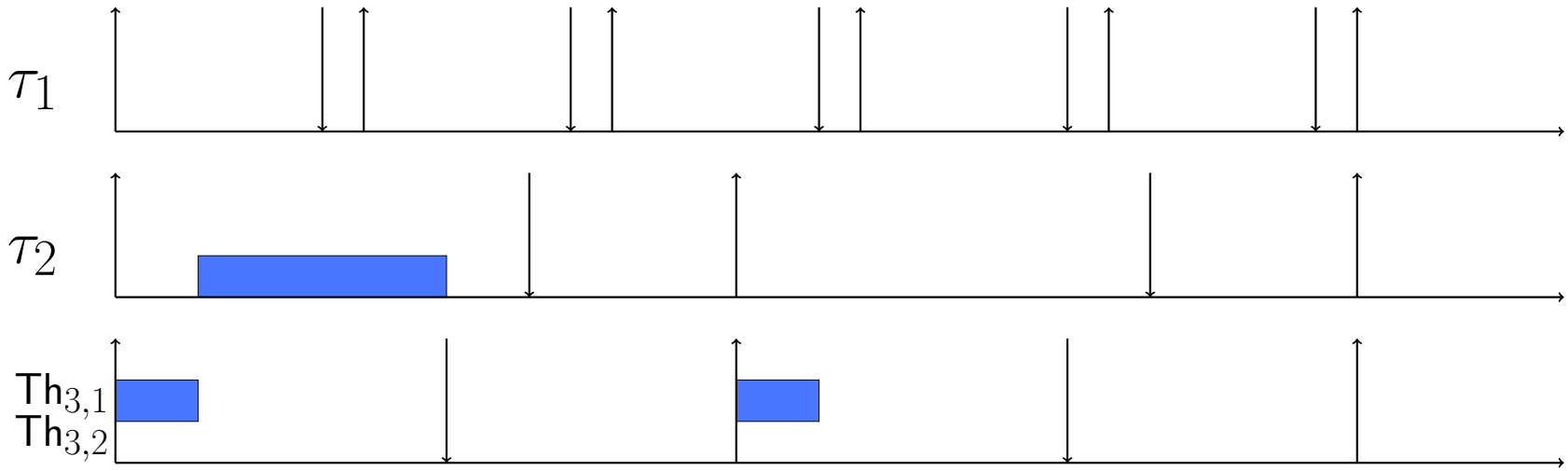
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



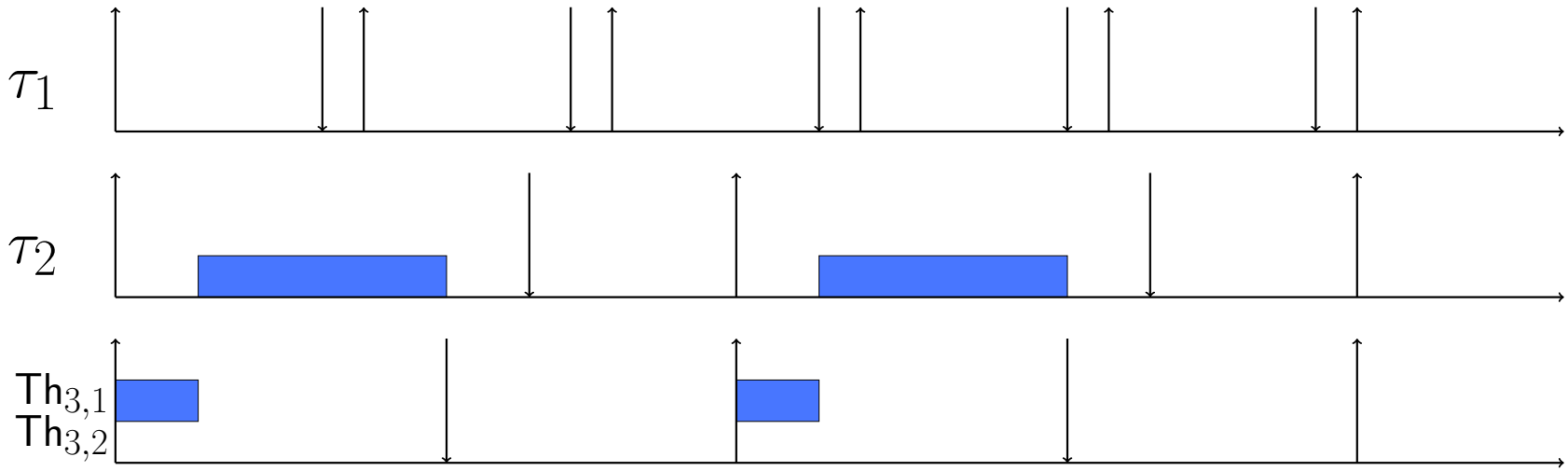
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



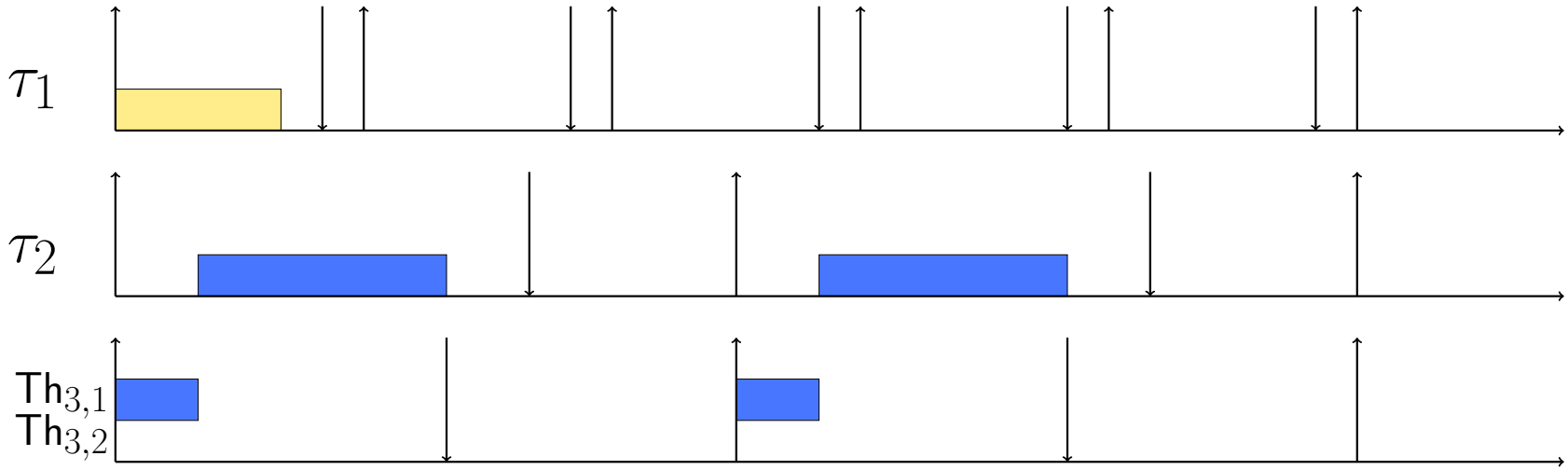
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



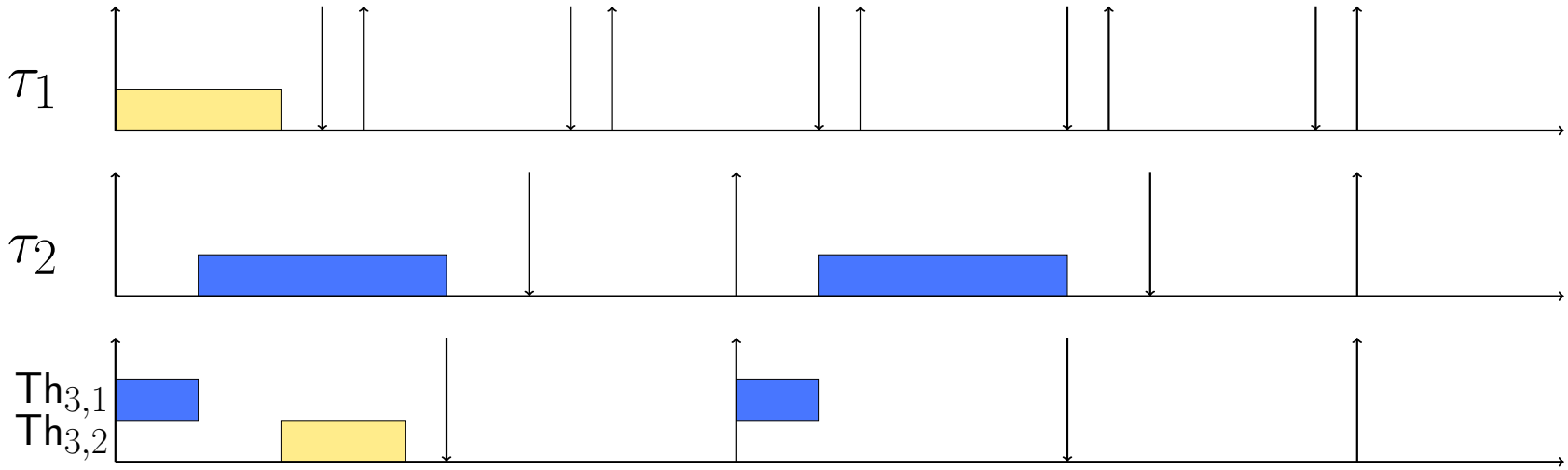
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



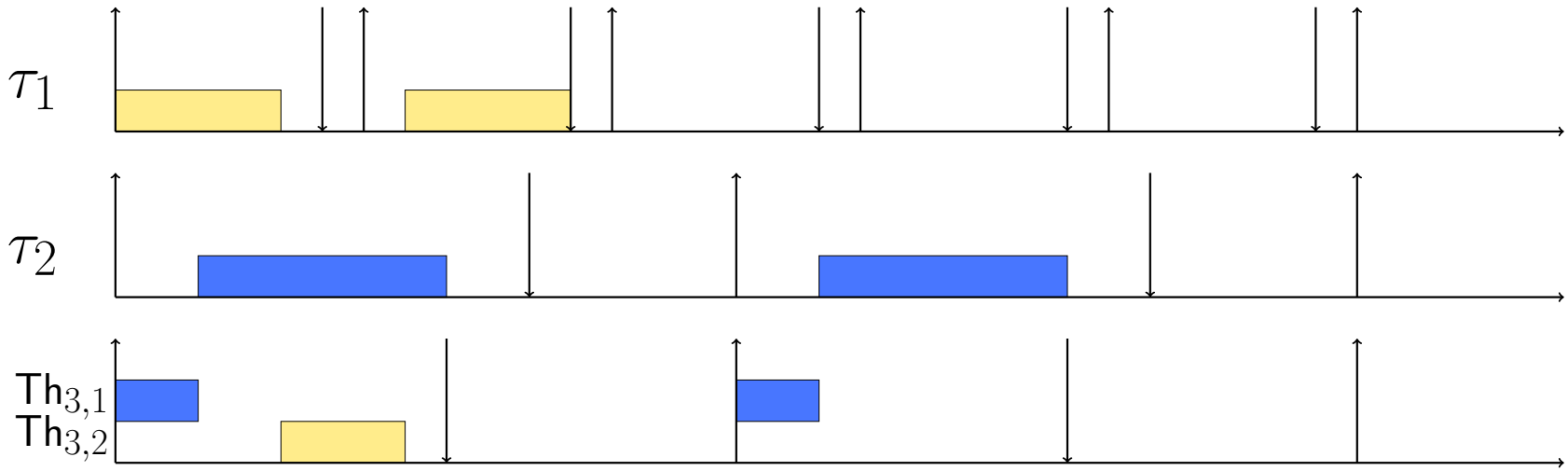
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



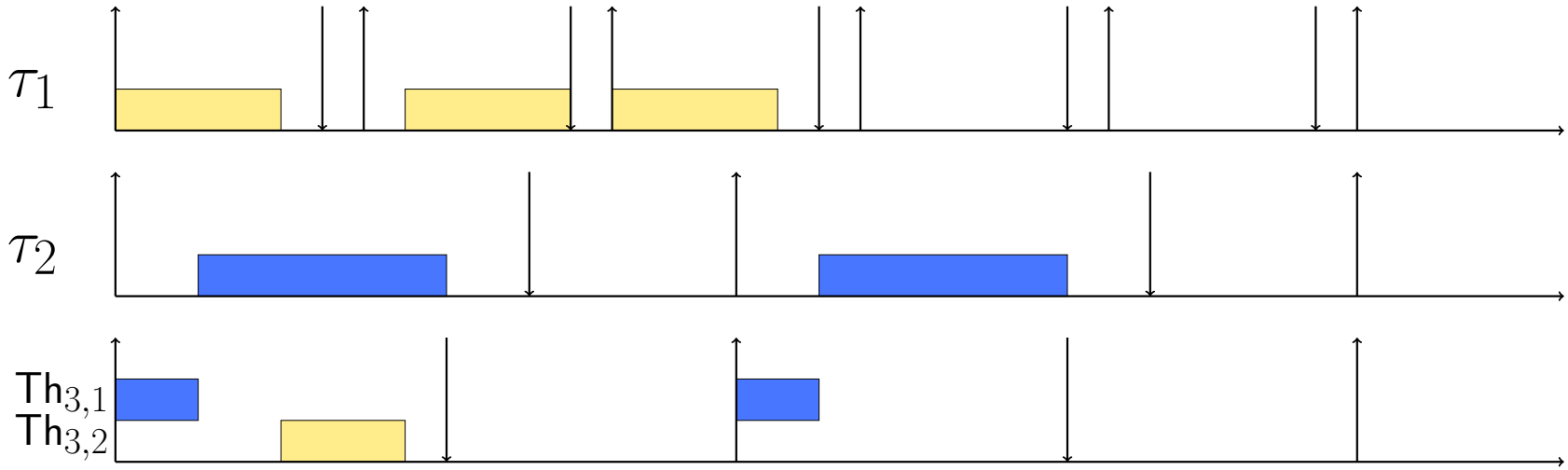
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



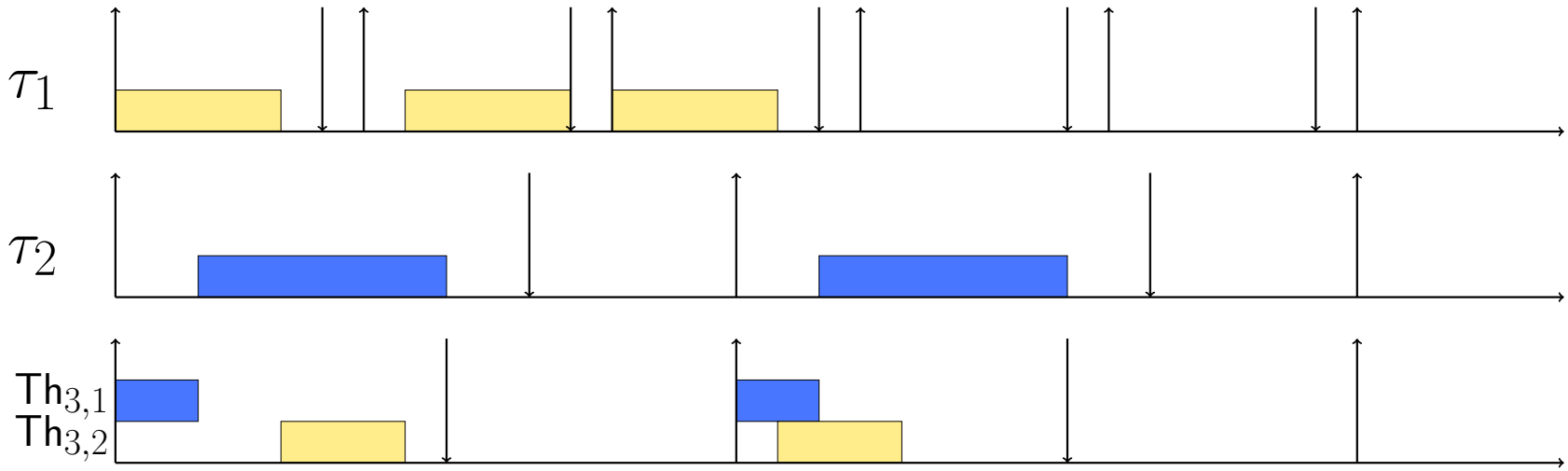
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



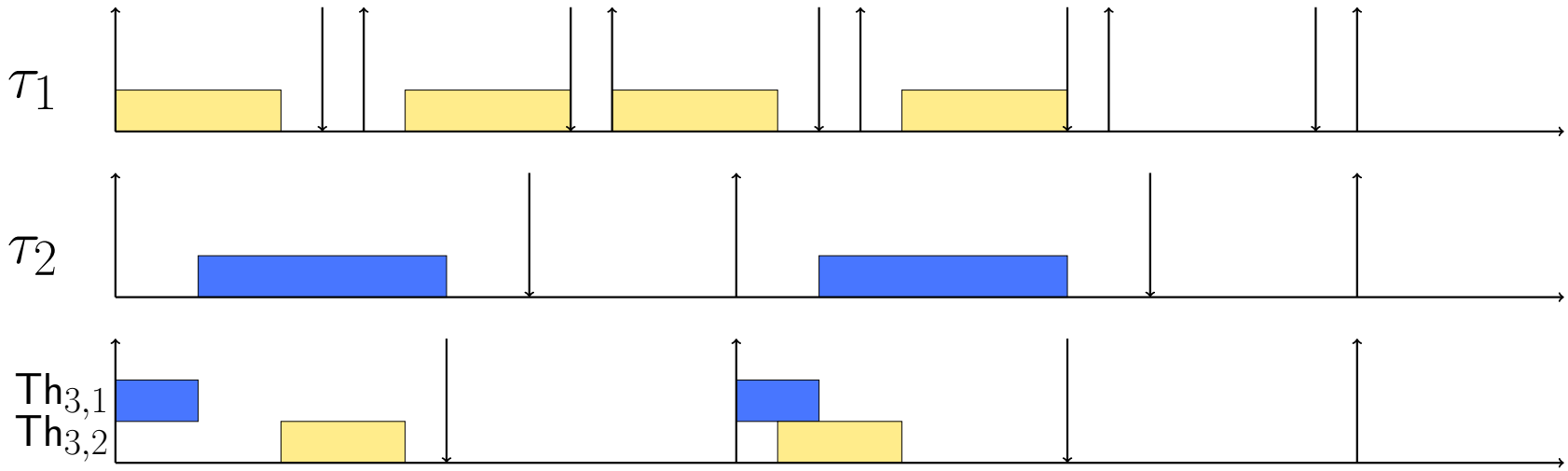
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



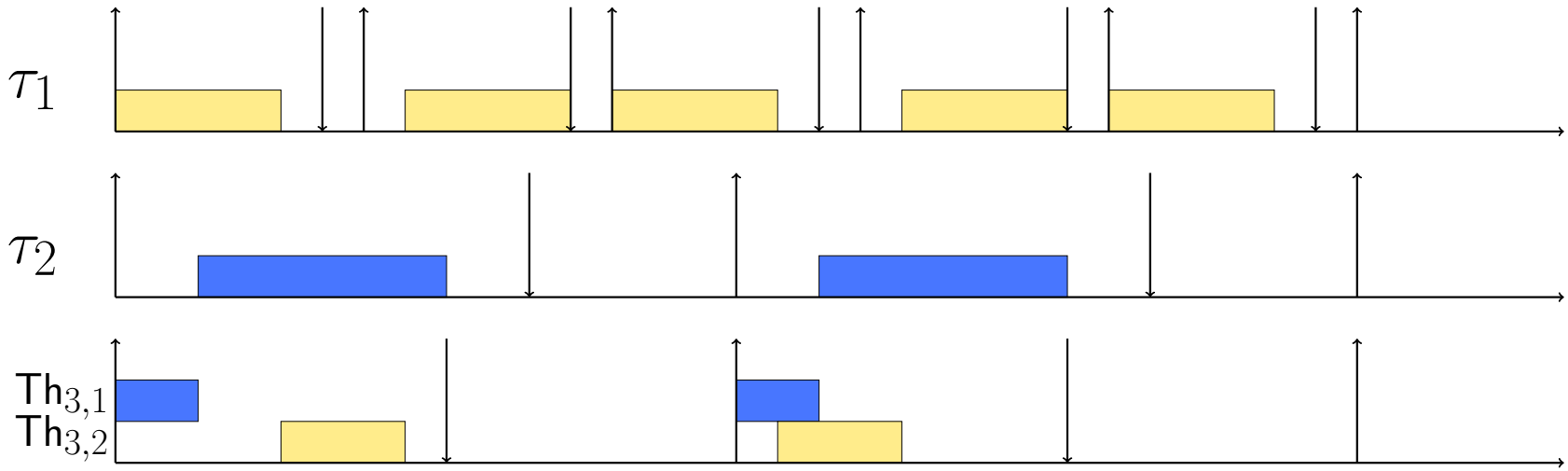
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



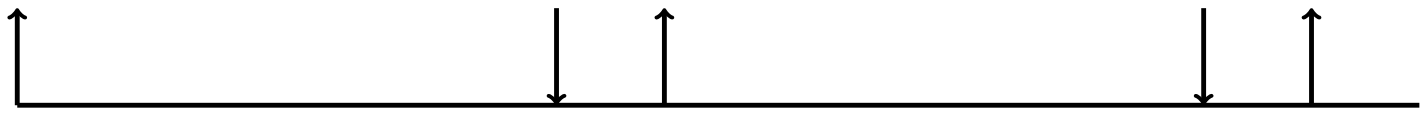
Un ordonnanceur EDF-partitionné parallèle

- On a 3 tâches à allouer sur 2 cœurs.
- Algorithme d'ordonnancement : Earliest Deadline First.
- Allocation : $\tau_1, \tau_3 \Rightarrow$ cœur jaune, $\tau_2 \Rightarrow$ cœur bleu
- "Enlever" 2 unités de temps de τ_3 et les allouer sur le cœur bleu



Parallélisation et la sélection de fréquence

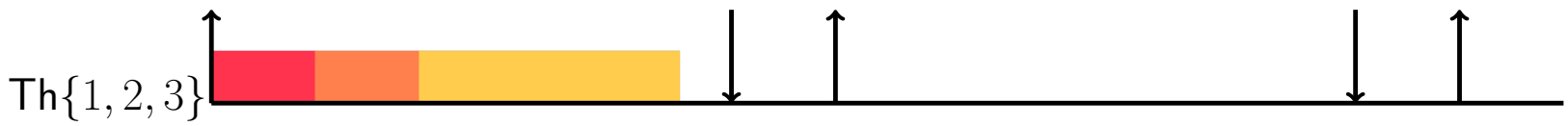
Soit τ une tâche qui peut être décomposée en 3 threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 1$



Puissance $\propto$ fréquence^3

Parallélisation et la sélection de fréquence

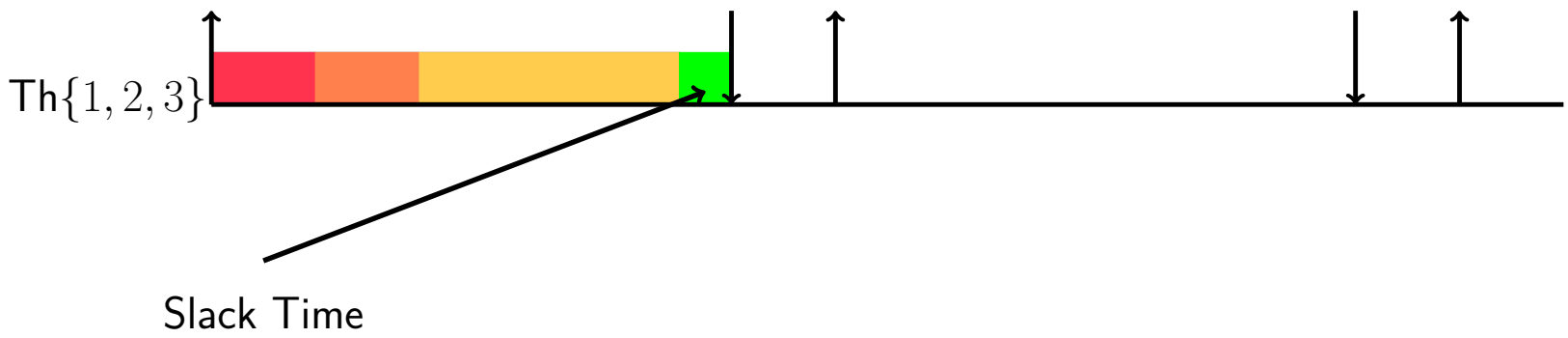
Soit τ une tâche qui peut être décomposée en 3 threads $\mathsf{Th}_1, \mathsf{Th}_2, \mathsf{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 1$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

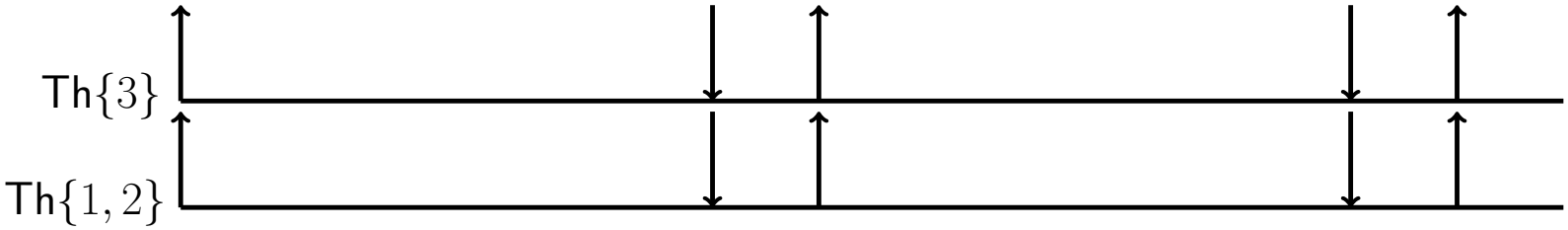
Soit τ une tâche qui peut être décomposée en 3 threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 1$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

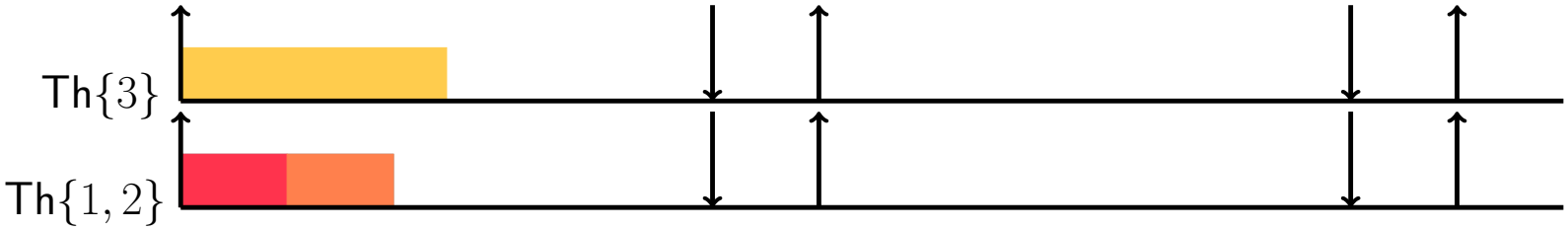
Soit τ une tâche qui peut être décomposée en 3 threads $\mathsf{Th}_1, \mathsf{Th}_2, \mathsf{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 1$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

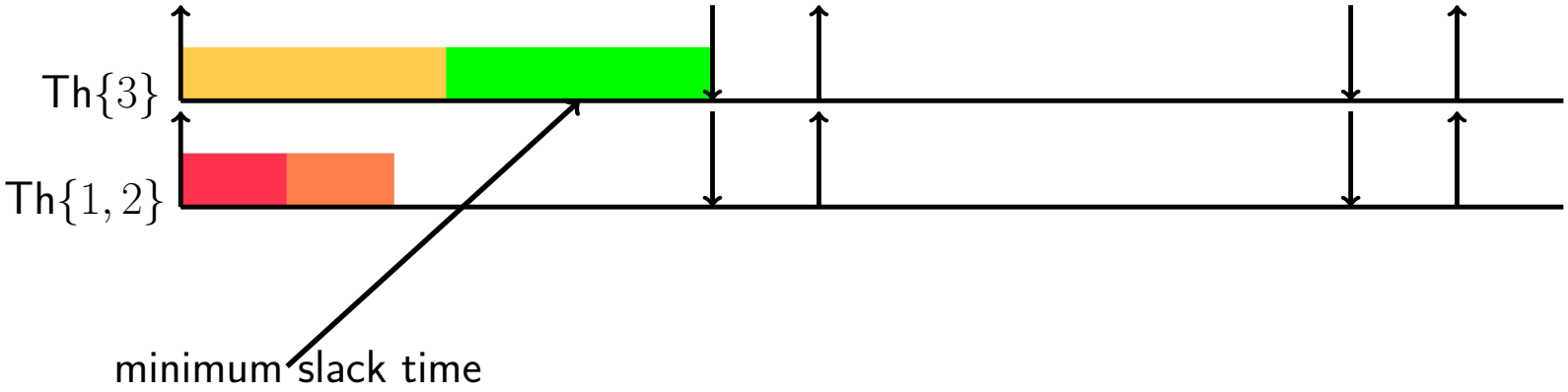
Soit τ une tâche qui peut être décomposée en 3 threads $\mathsf{Th}_1, \mathsf{Th}_2, \mathsf{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 1$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

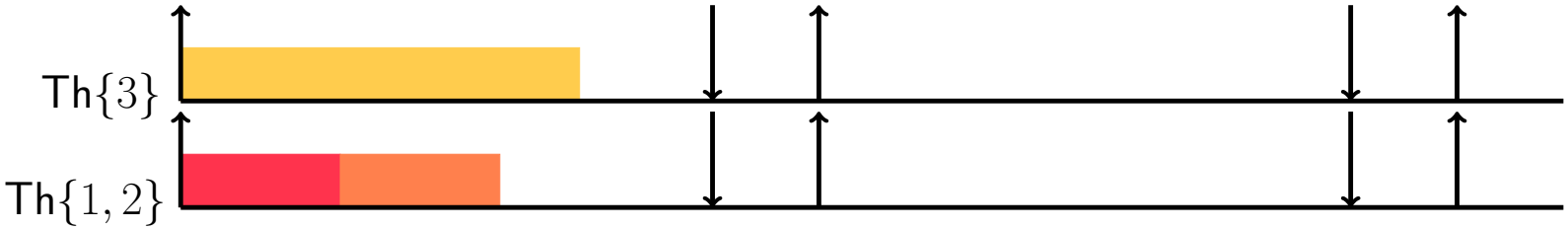
Soit τ une tâche qui peut être décomposée en 3 threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 1$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

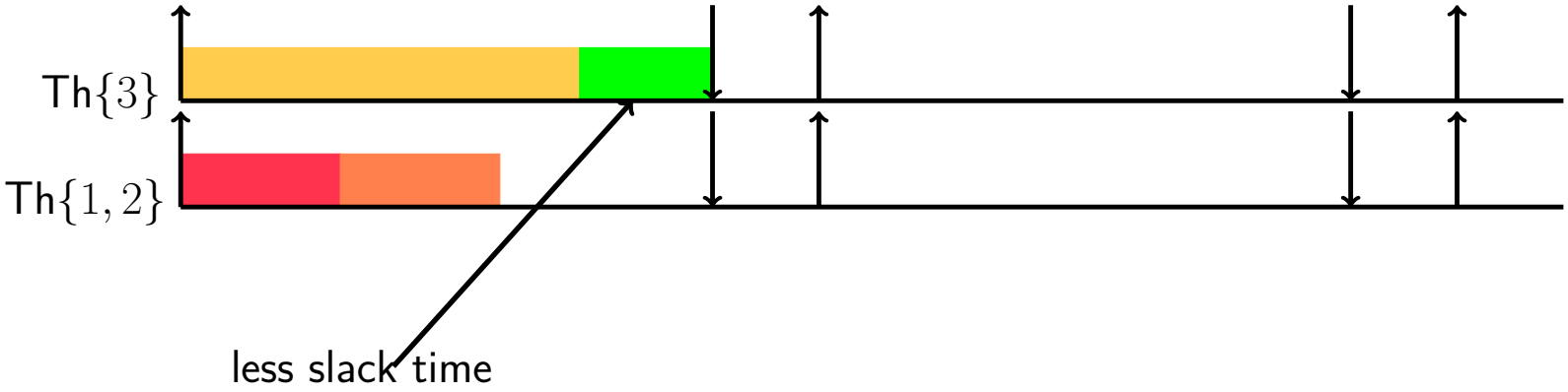
Soit τ une tâche qui peut être décomposée en 3 threads $\mathsf{Th}_1, \mathsf{Th}_2, \mathsf{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 0.75$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

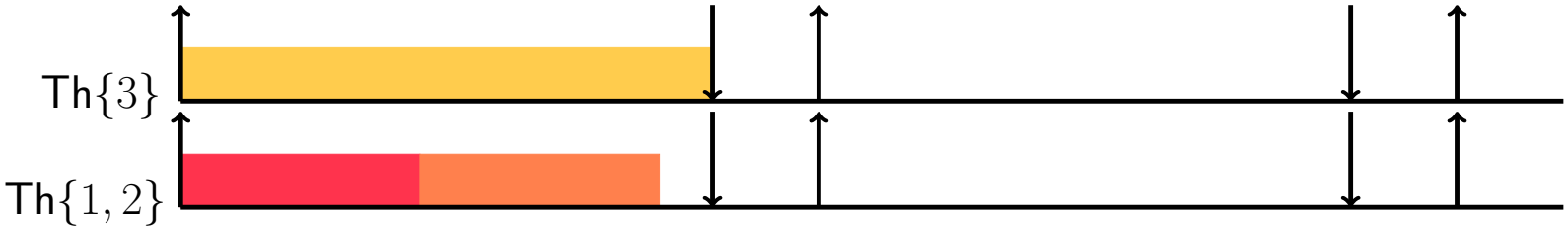
Soit τ une tâche qui peut être décomposée en 3 threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 0.75$



$$\text{Puissance} \propto \text{fréquence}^3$$

Parallélisation et la sélection de fréquence

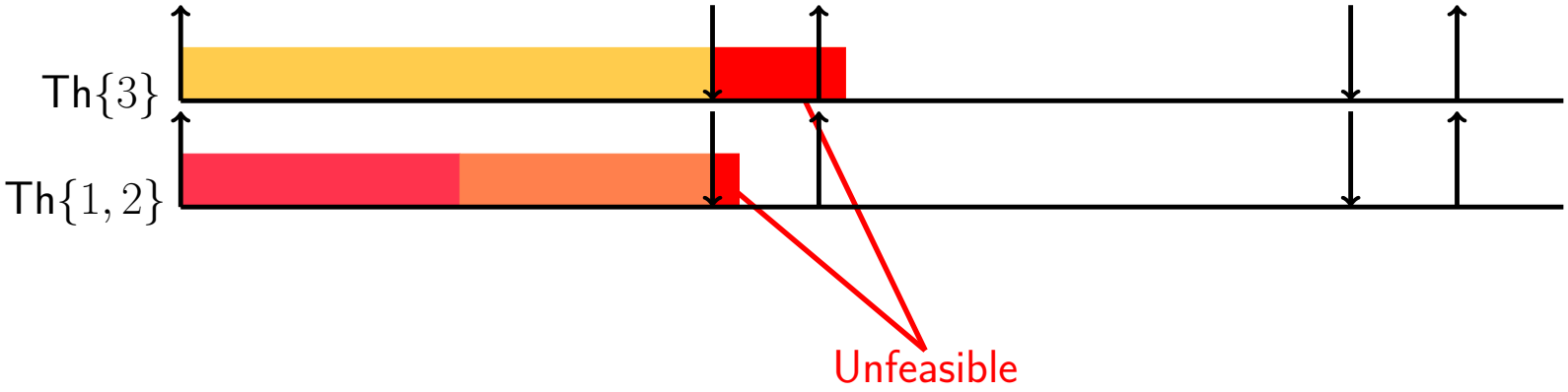
Soit τ une tâche qui peut être décomposée en 3 threads $\mathsf{Th}_1, \mathsf{Th}_2, \mathsf{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 0.50$



$$\text{Puissance} \propto \text{fréquence}^3$$

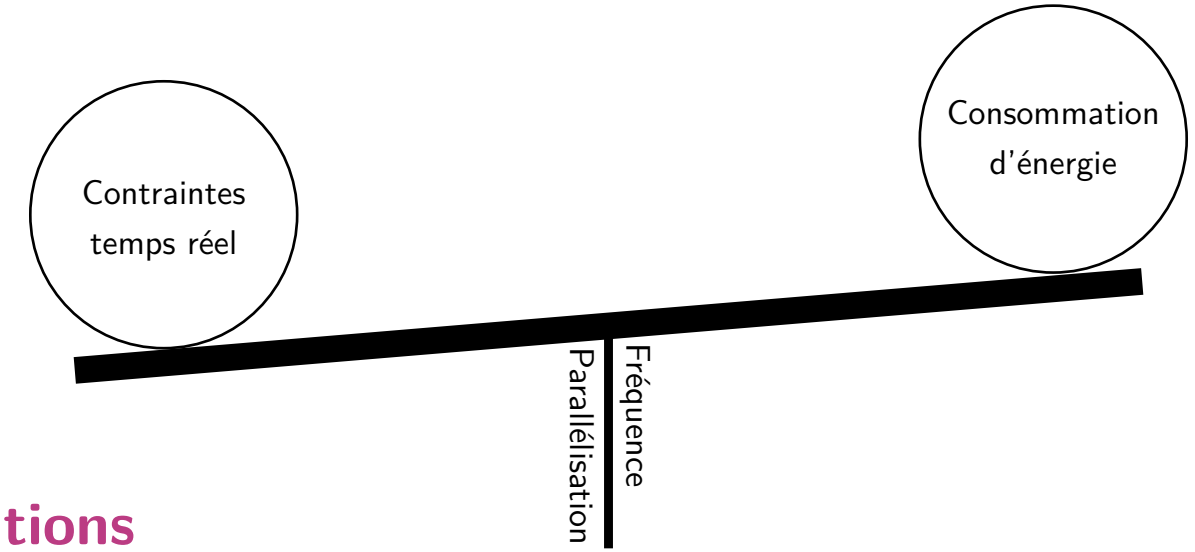
Parallélisation et la sélection de fréquence

Soit τ une tâche qui peut être décomposée en 3 threads $\mathrm{Th}_1, \mathrm{Th}_2, \mathrm{Th}_3$, qui ont des temps d'exécution de 2,2,5 sur des coeurs qui opèrent à la vitesse $s = 0.35$



$$\text{Puissance} \propto \text{fréquence}^3$$

Contributions



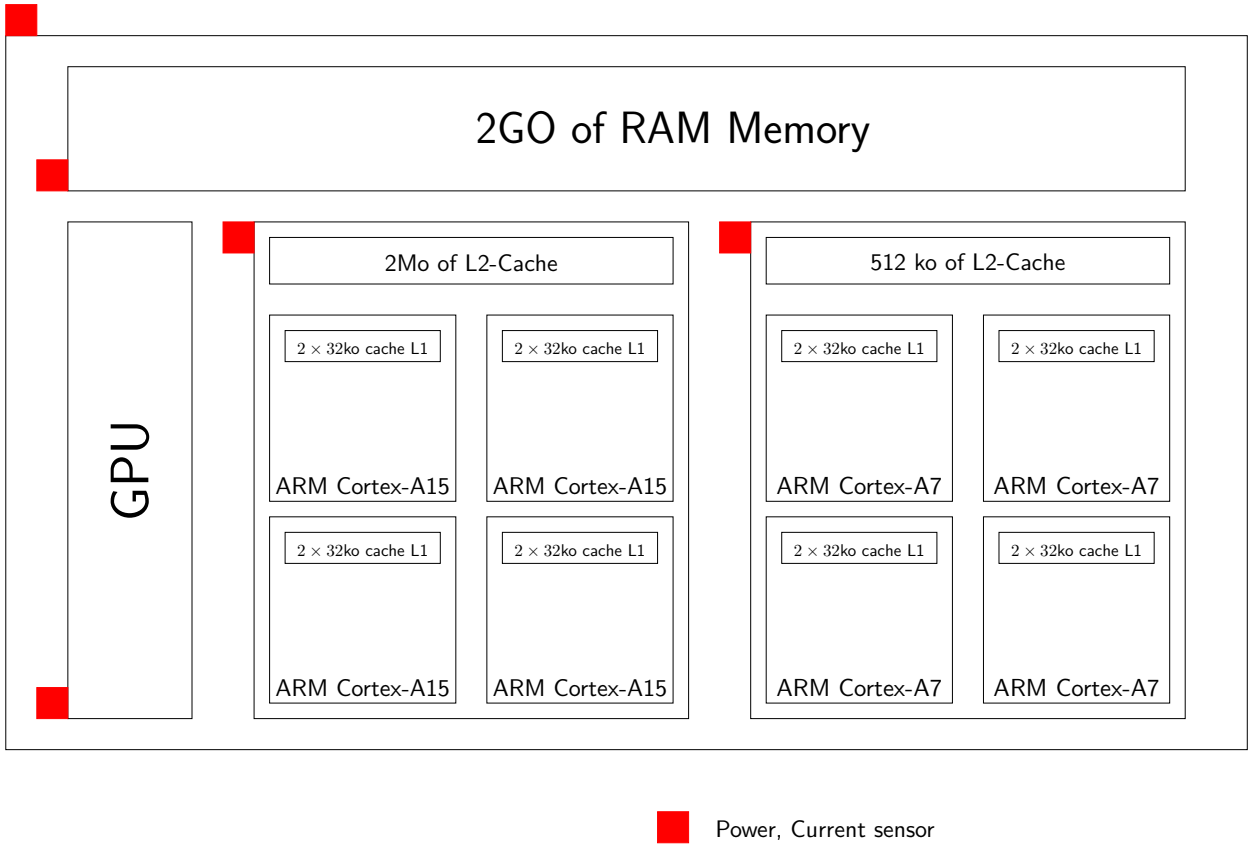
Contributions

- L'interaction entre fréquence, temps d'exécution et énergie
- La modélisation du parallélisme de tâches temps réel par les cut-points
- Expression du comportement dynamique des tâches temps réel parallèles par les digraphes

Plan

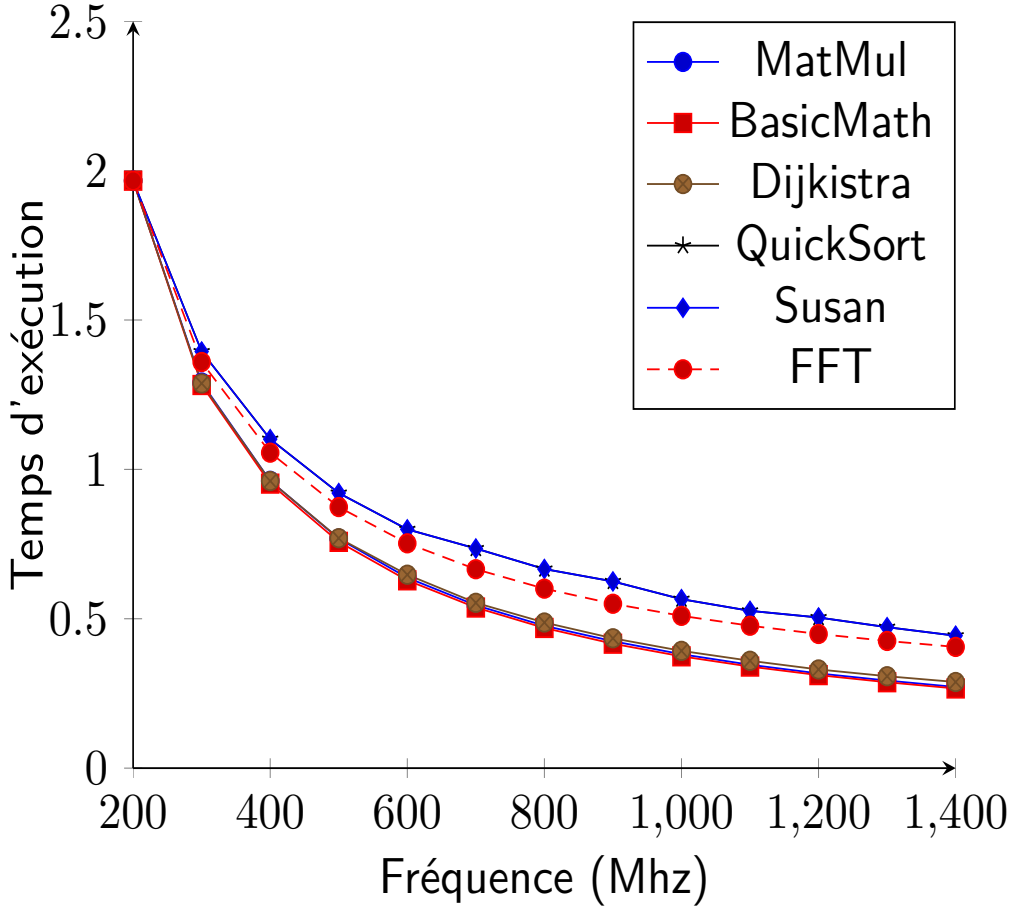
Performances d'une plateforme hétérogène

- **But :** Concevoir un modèle de "temps" et de consommation d'énergie
  - Exécuter différentes tâches récurrentes avec des priorités temps réel sur une architecture hétérogène.



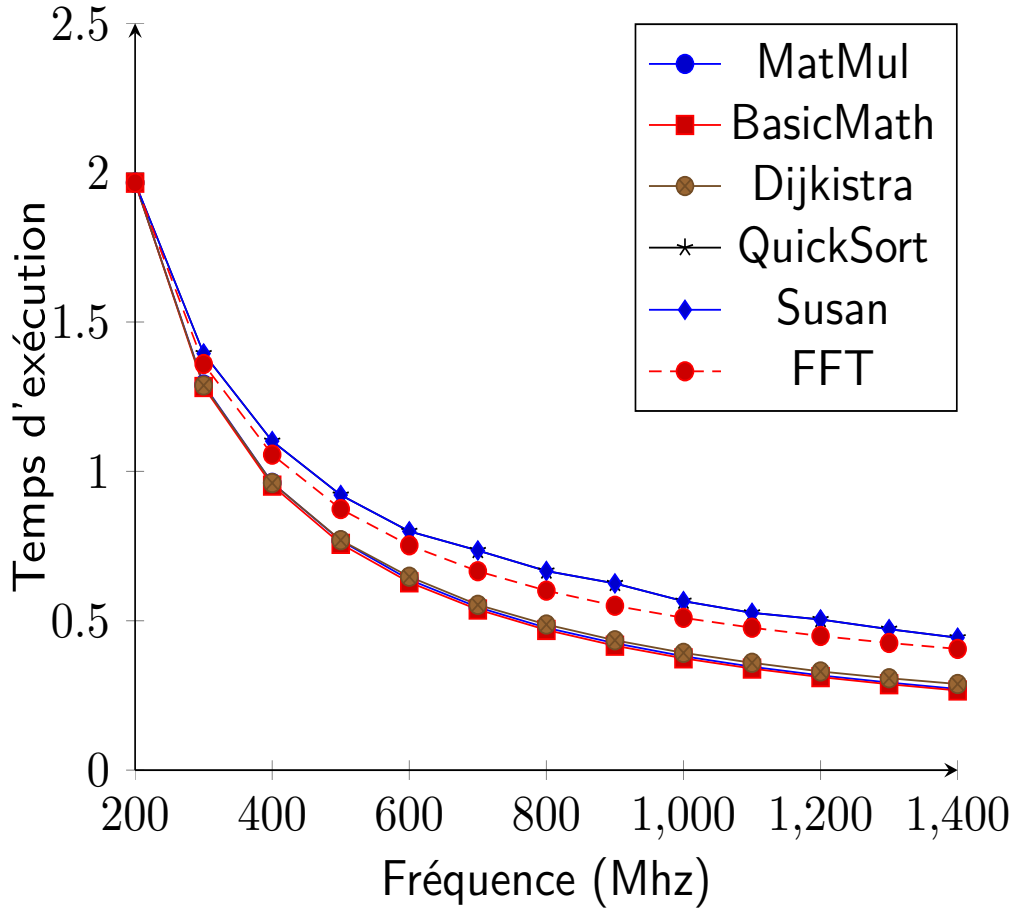
Modèle de temps d'exécution

- Un thread sur un cœur little



Modèle de temps d'exécution

- Un thread sur un cœur little

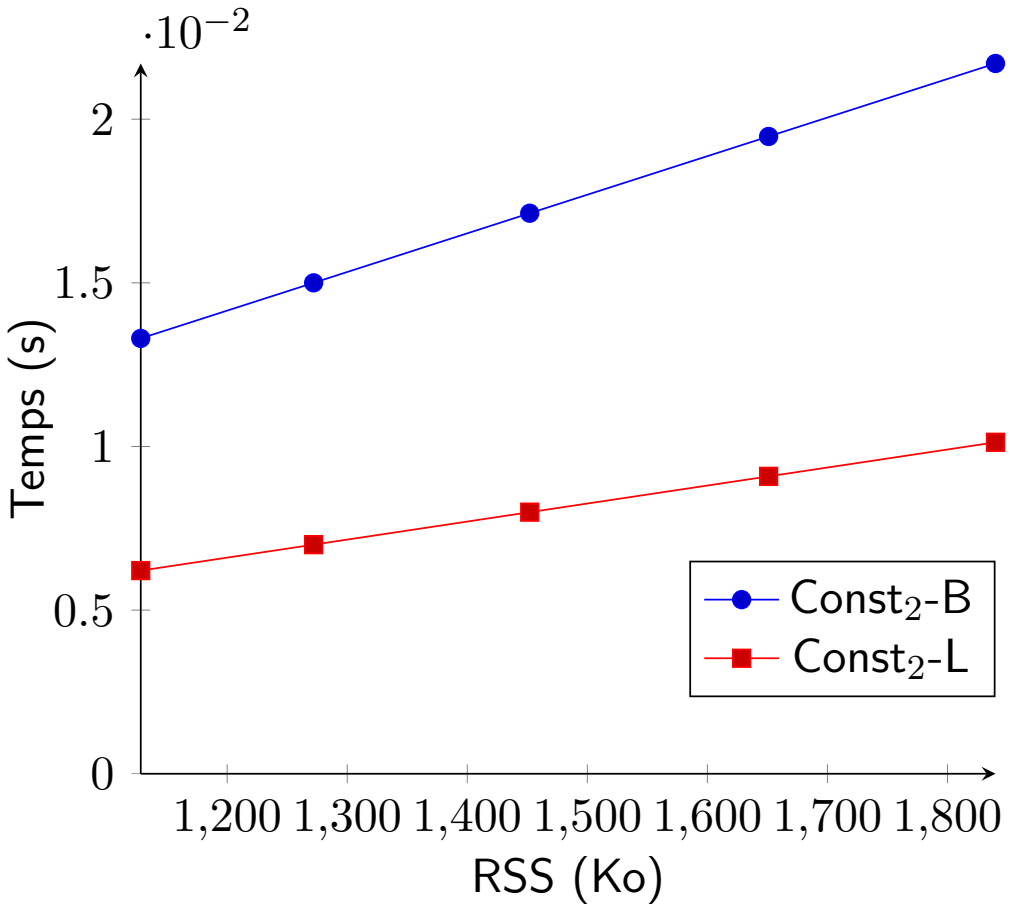


- Le temps d'exécution est une fonction de la tâche et de la fréquence
- Régression non-linéaire : $\mathcal{C}(\mathsf{f}_{op}) = \frac{\mathsf{Const}_1}{\mathsf{f}_{op}} + \mathsf{Const}_2$

Le modèle de temps d'exécution

- La taille des données traitées est changée

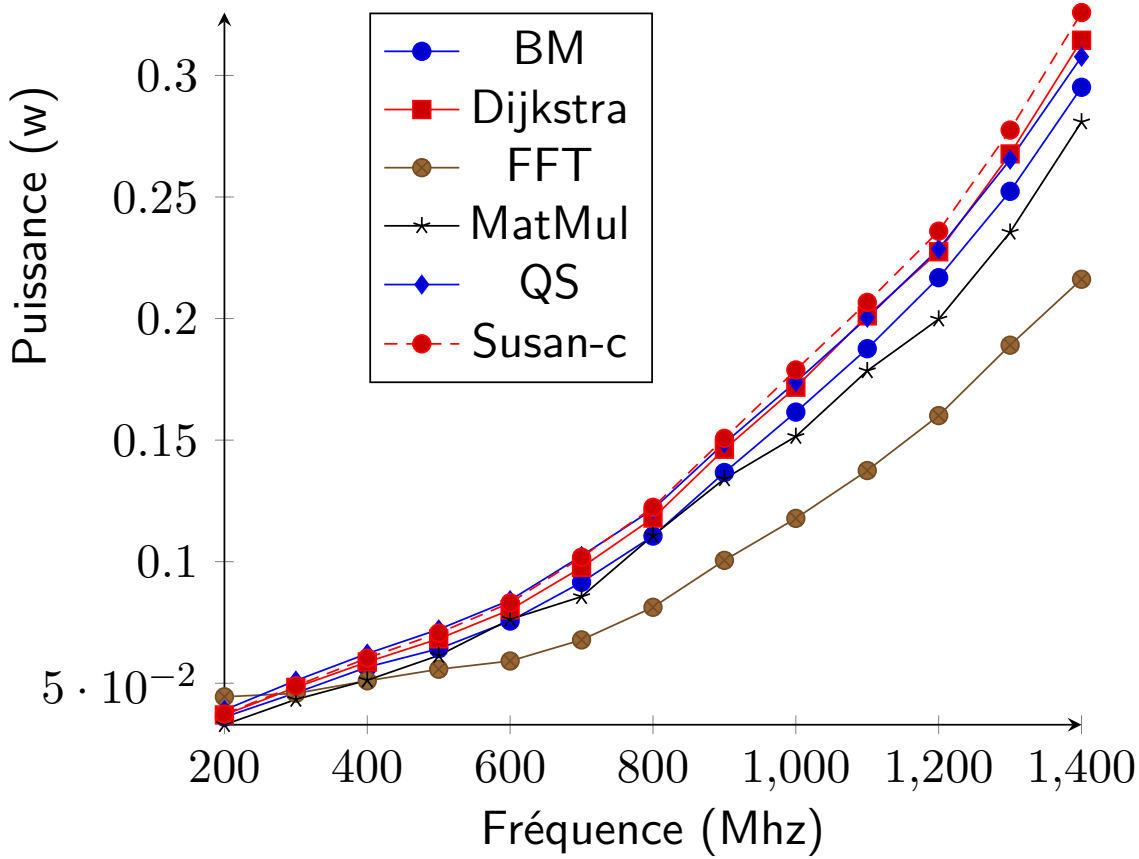
$$\mathcal{C}(\mathsf{f}_{op}) = \frac{\mathsf{Const}_1}{\mathsf{f}_{op}} + \mathsf{Const}_2$$



- *Const*$_2$ représente le temps d'accès mémoire

Le modèle de puissance

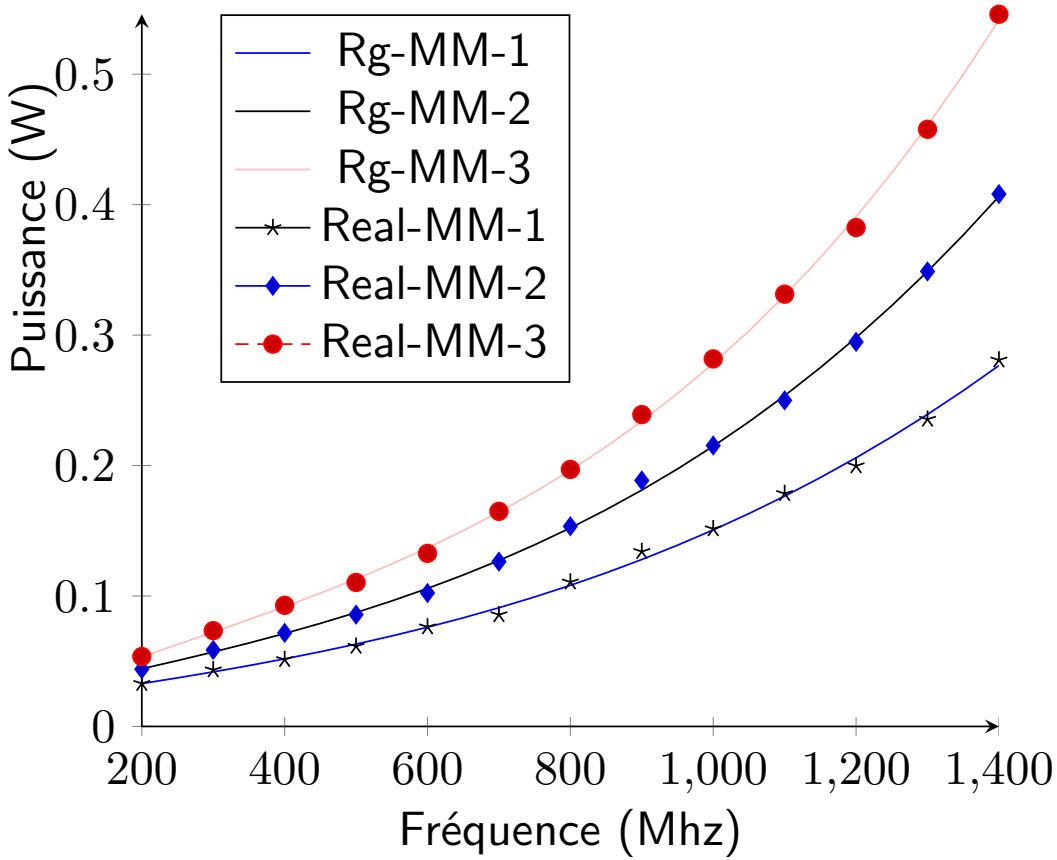
- Un thread par cœur



- La puissance dissipée dépend de la tâche

Le modèle de la puissance : Régression VS valeurs réelles

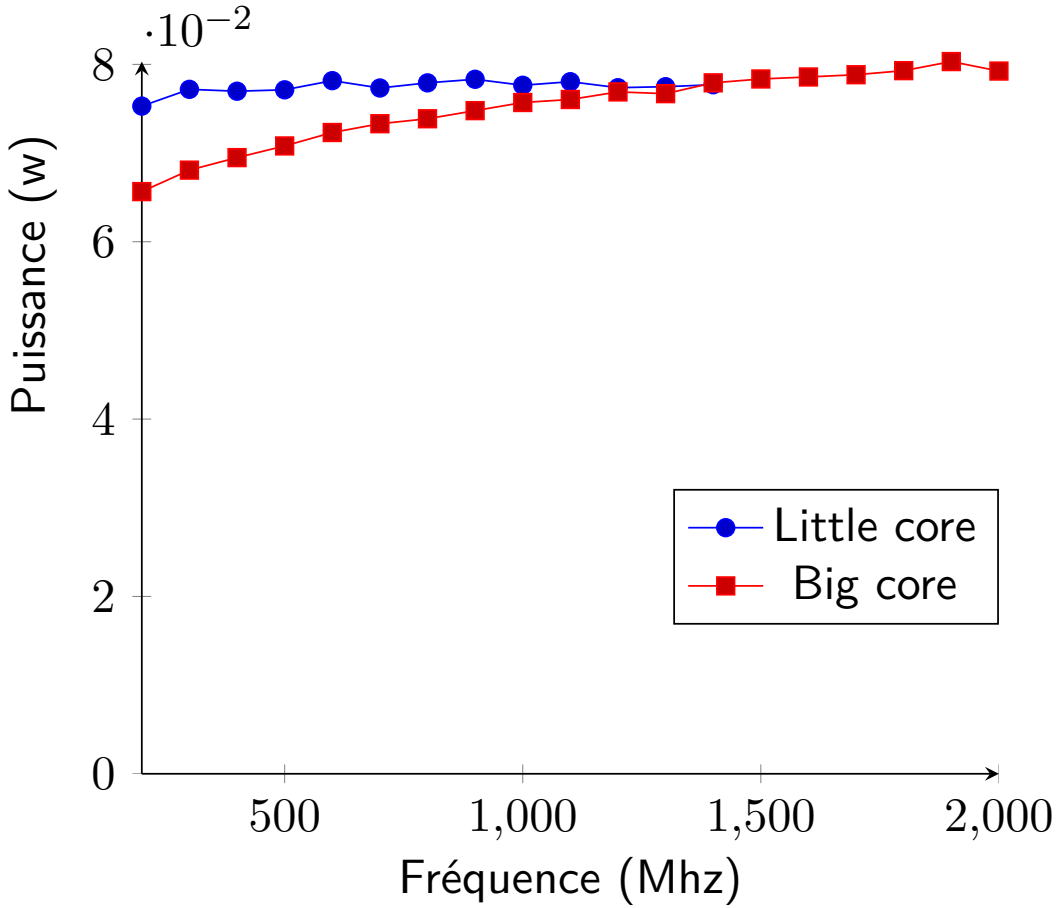
- Régression polynomiale de 3^e degré est appliquée
- La régression est très exacte



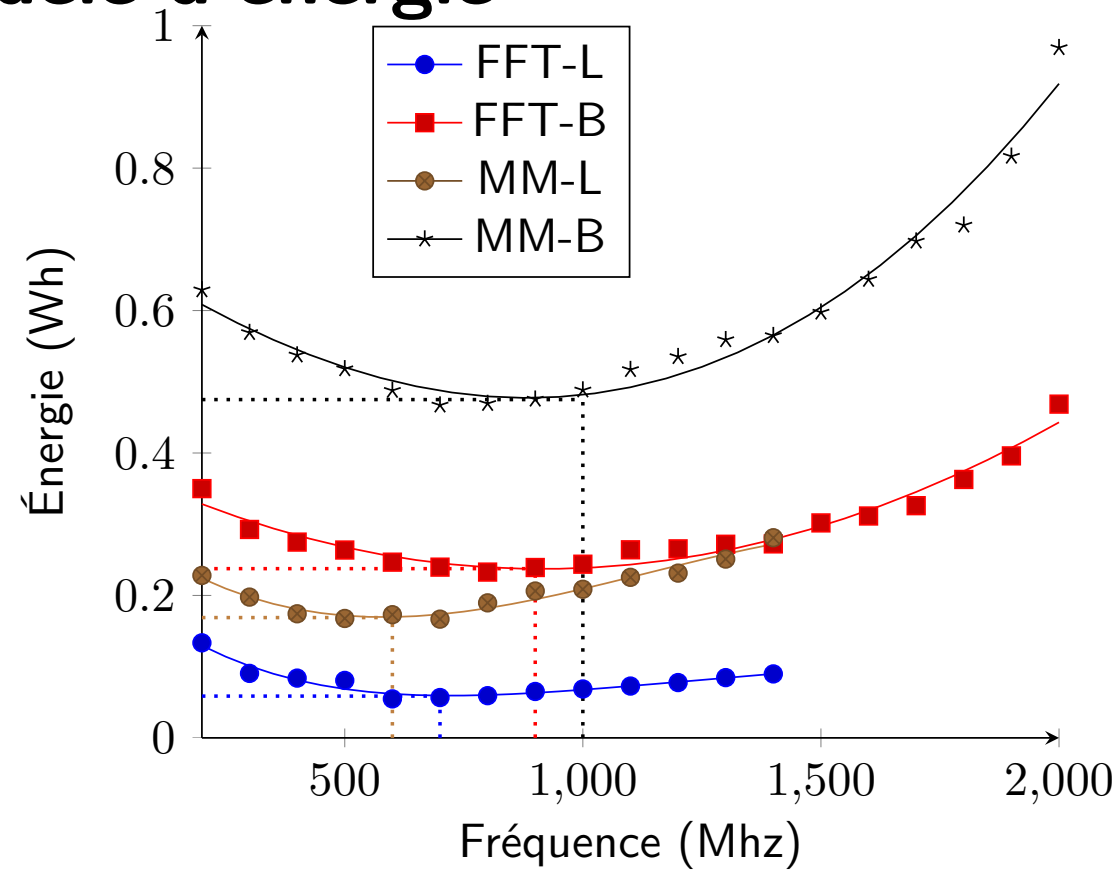
- La puissance dissipée par 2 threads n'est pas $\times 2$ la puissance dissipée par 1 thread.

Le modèle de puissance : La mémoire

- Quand un thread est alloué sur un cœur little, la mémoire consomme plus que s'il est alloué sur un coeur big
  - Cache-L2 plus petit $\Rightarrow$ plus cache-miss $\Rightarrow$ plus d'accès mémoire.
- La mémoire consomme plus que 20% de la consommation totale d'un coeur little



Le modèle d'énergie



- Augmenter la fréquence "peut" aider à réduire la consommation d'énergie jusqu'à une certaine fréquence (*effective*).
- La fréquence effective varie d'une tâche à une autre

Que retenir de ces expérimentations

La puissance dissipée dépend :

- Du type du coeur où la tâche est allouée
- De la fréquence du coeur
- De la tâche elle-même

Quelques remarques :

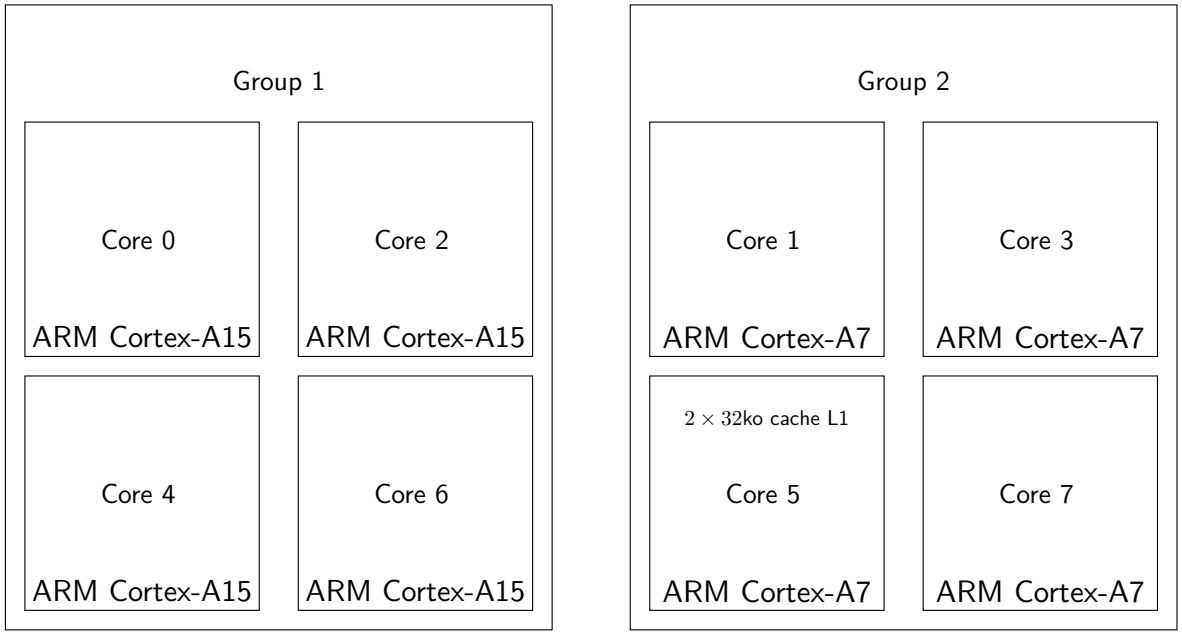
- Deux threads de la même tâche dissipent la même puissance.
- L'énergie statique dépend du voltage qui peut dépendre de la fréquence.
- La consommation d'énergie de la mémoire est importante.
- La fréquence effective est propre à chaque tâche.

Plan

Le modèle d'architecture

Une architecture hétérogène est modélisée par :

- Un ensemble de G groupes
- Chaque groupe $g \in G$ est composé d'un ensemble de coeurs
- Les coeurs du même groupe opèrent sur la même fréquence.

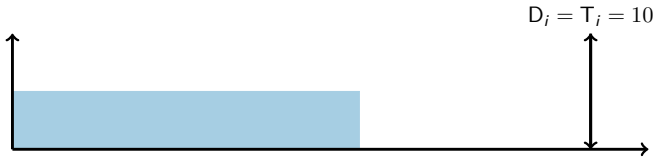


Modèle de tâches

Modèle de tâches

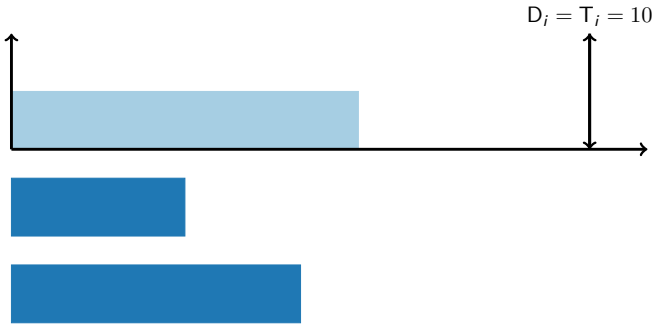
$$\mathcal{C}(f_{op}) = \frac{Const_1}{f_{op}} + Const_2$$

Modèle de tâches



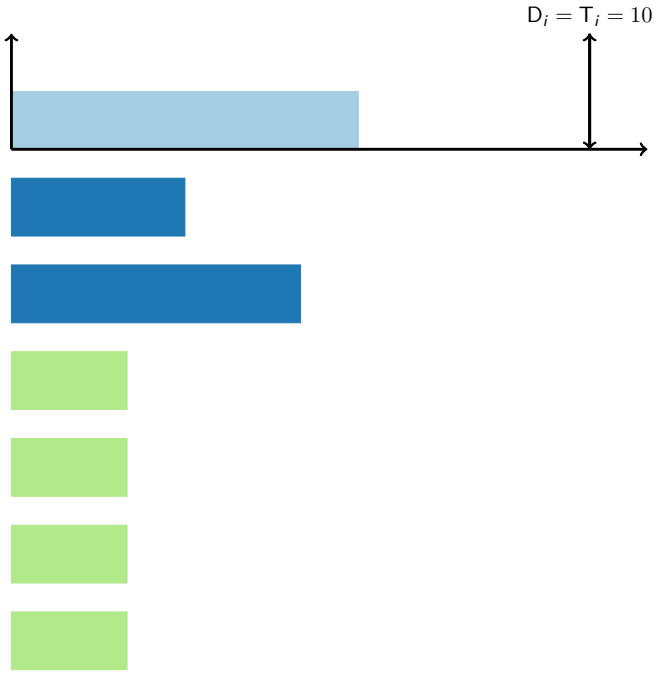
$$\mathcal{C}(f_{op}) = \frac{Const_1}{f_{op}} + Const_2$$

Modèle de tâches



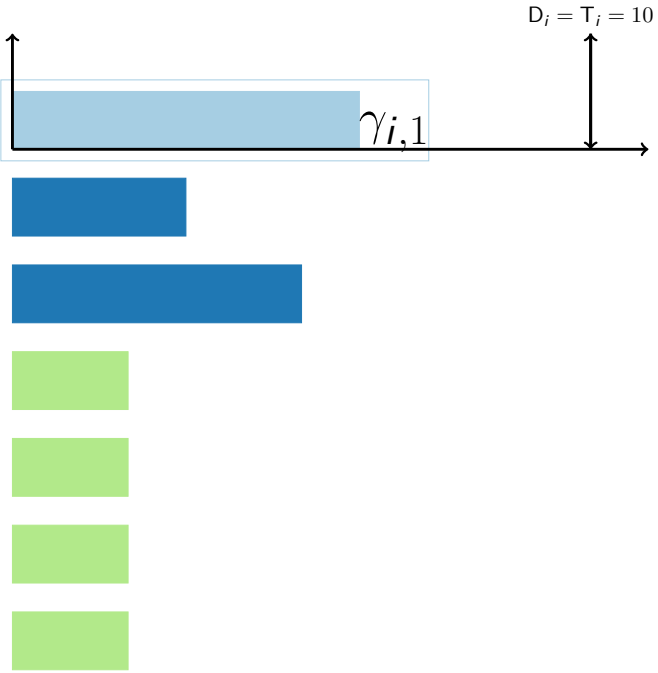
$$\mathcal{C}(f_{op}) = \frac{Const_1}{f_{op}} + Const_2$$

Modèle de tâches



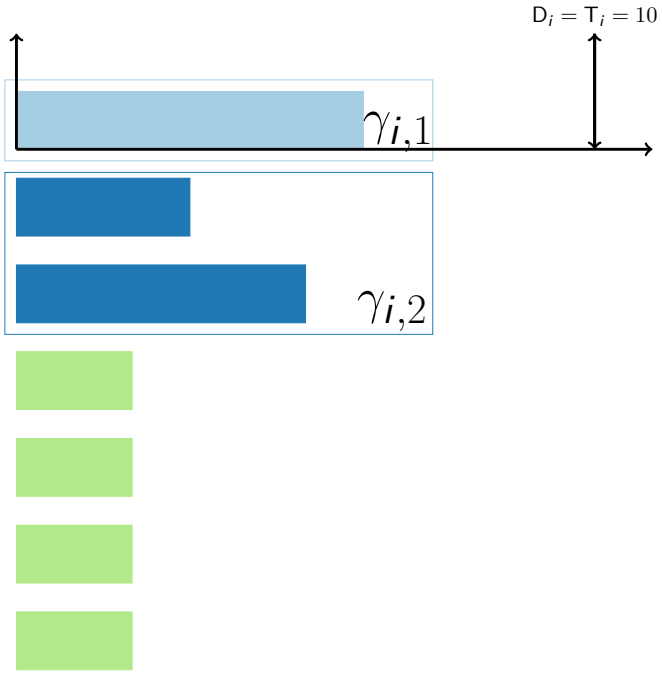
$$\mathcal{C}(f_{op}) = \frac{Const_1}{f_{op}} + Const_2$$

Modèle de tâches



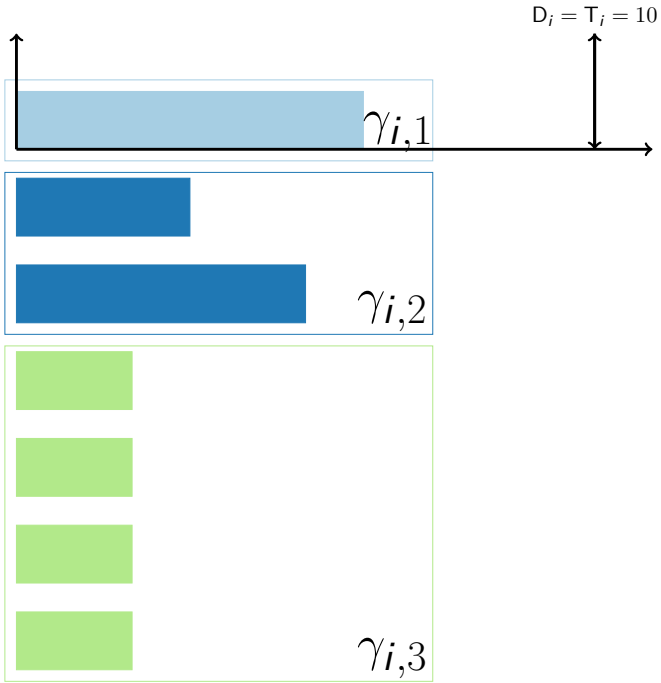
$$\mathcal{C}(f_{op}) = \frac{\text{Const}_1}{f_{op}} + \text{Const}_2$$

Modèle de tâches



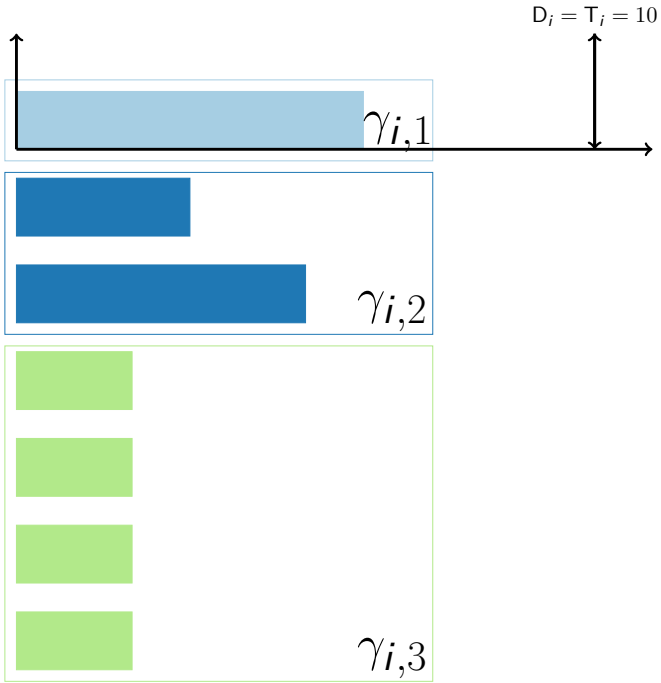
$$\mathcal{C}(f_{op}) = \frac{Const_1}{f_{op}} + Const_2$$

Modèle de tâches



$$\mathcal{C}(f_{op}) = \frac{Const_1}{f_{op}} + Const_2$$

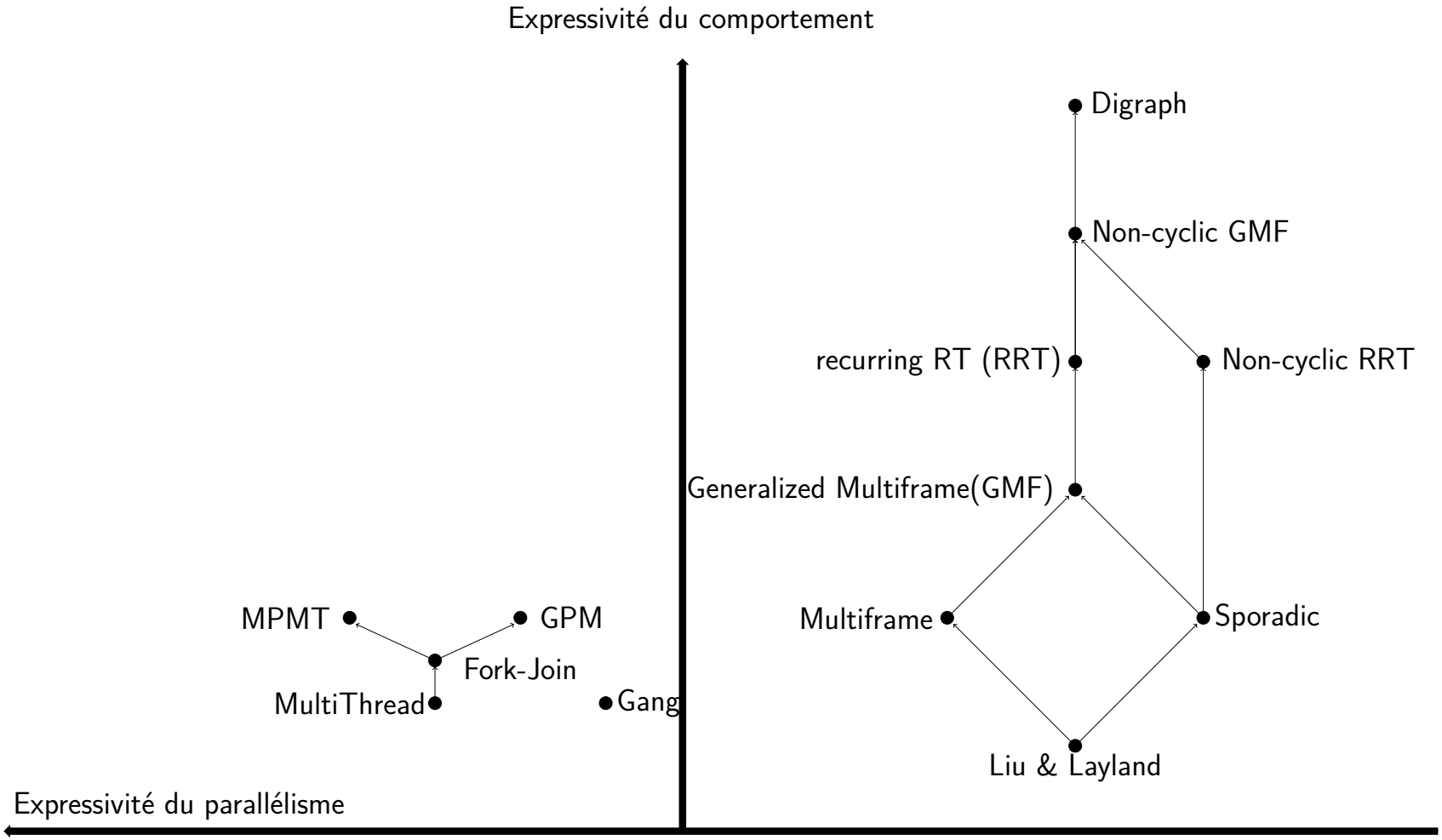
Modèle de tâches



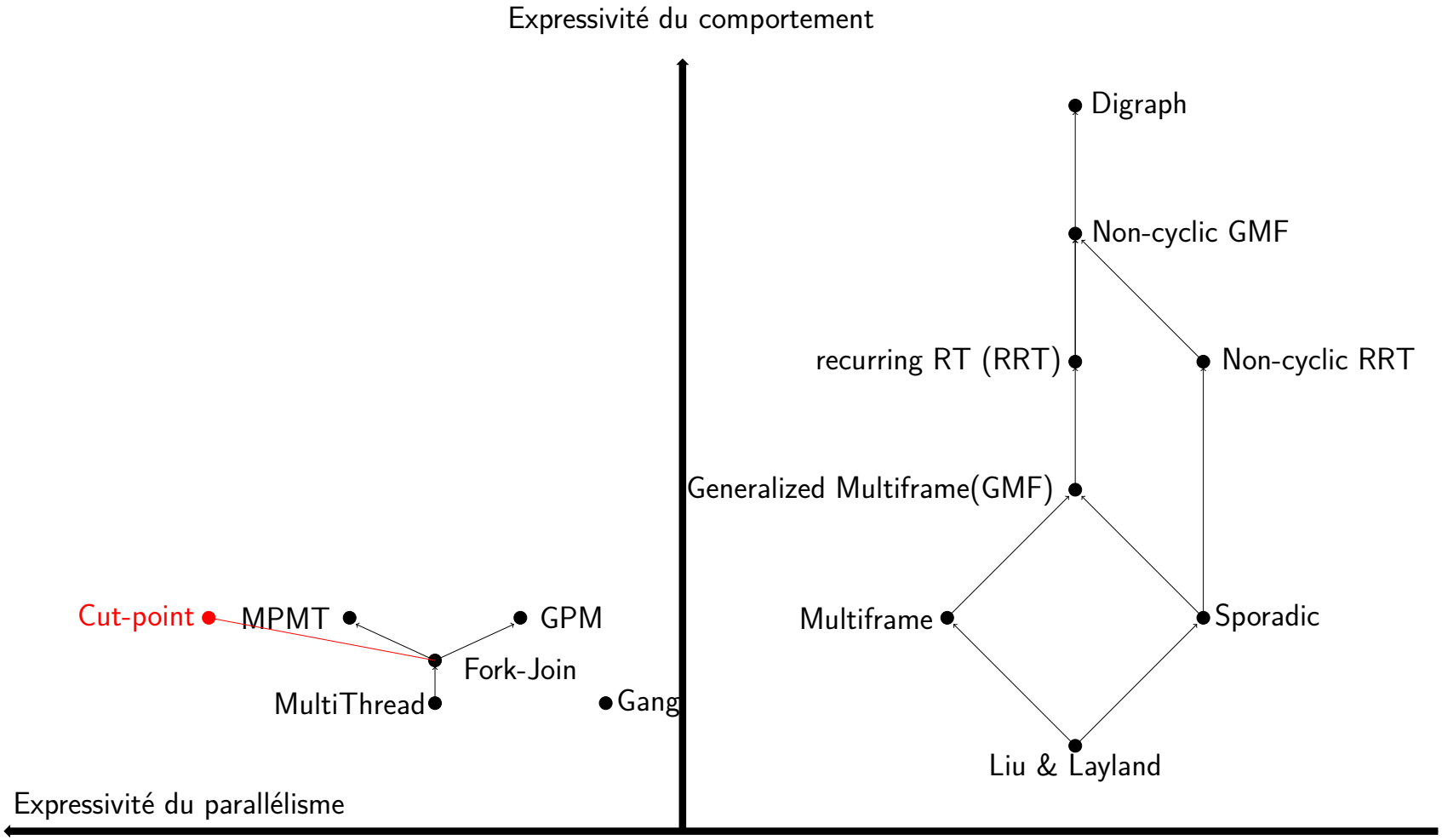
$$\mathcal{C}(f_{op}) = \frac{\text{Const}_1}{f_{op}} + \text{Const}_2$$

- Le temps d'exécution de chaque thread est caractérisé par : $\text{Const}_1^g, \text{Const}_2^g$
- $\sum_z C_{i,k,z} \geq C_{i,1,1}, \forall i, k$
- $\vec{\epsilon}$ un vecteur des coefficients du polynômes de la consommation d'énergie

Modèle cut-point et modèles de tâches



Modèle cut-point et modèles de tâches



Problème à résoudre

Nous avons :

- Un ensemble de tâches modélisées par CPM.
- Une plateforme avec un ensemble de coeurs hétérogènes

Objectif

- Allouer les tâches aux coeurs
- Sélectionner la fréquence minimale en hors-ligne pour chaque groupe.

Contraintes

- Toutes les échéances doivent être respectées.

Solution Exacte : INLP

- Partitionné : migration difficile à gérer
- Solution exacte : énumération des combinaisons par la programmation en nombre entier

Formulation en INLP

- $x_{i,j,k,z}$ définit l'allocation du thread $\mathrm{Th}_{i,k,z}$ au coeur j
- f^g la fréquence opérationnelle du groupe g
- La fonction objectif est non-linéaire : $minE = \cdots x_{i,j,k,z} \ \frac{C_{i,k,z}}{f^g} \cdots$

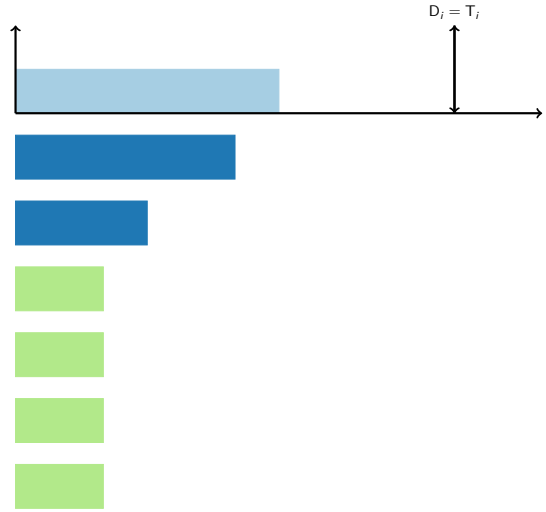
Exemple : 5 tâches, 5 cut-points , 8 threads, 8 coeurs, hyper period = 1000$\rightarrow$ 2 568 038 contraintes

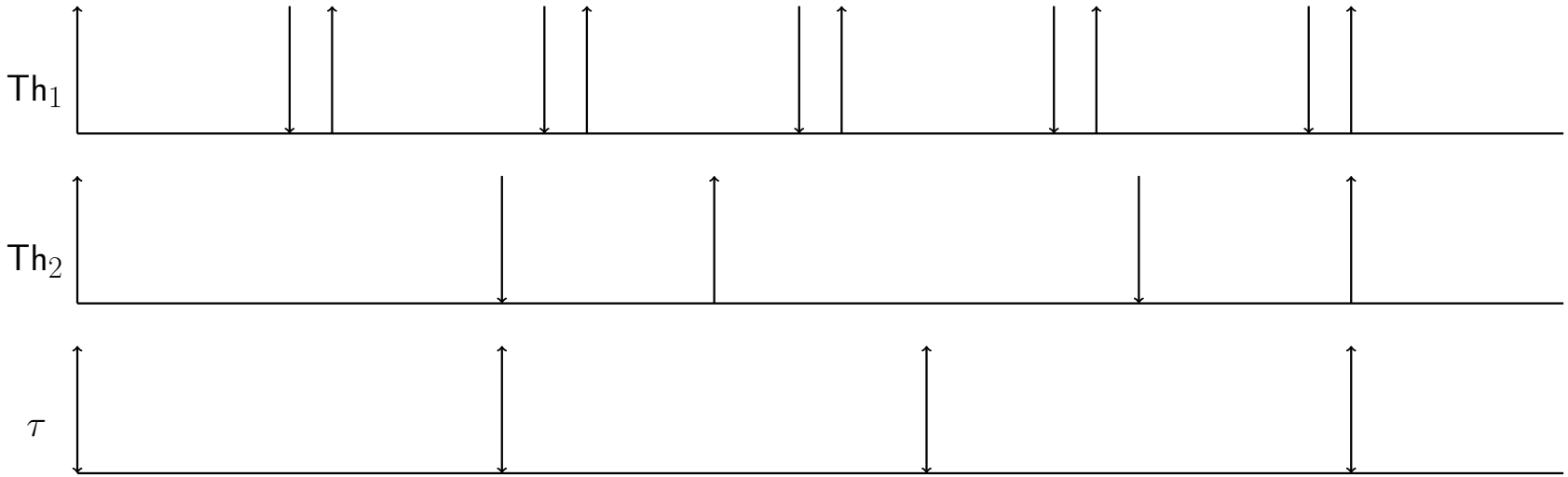
Pas facile à résoudre

La résolution peut prendre plusieurs heures pour un problème de petite taille.

L'heuristique de partitionnement

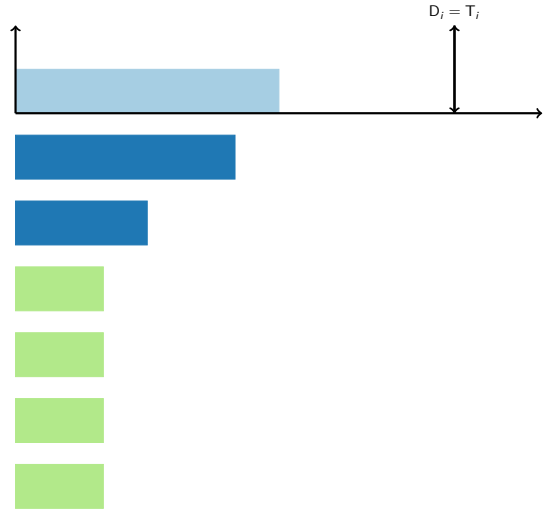
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

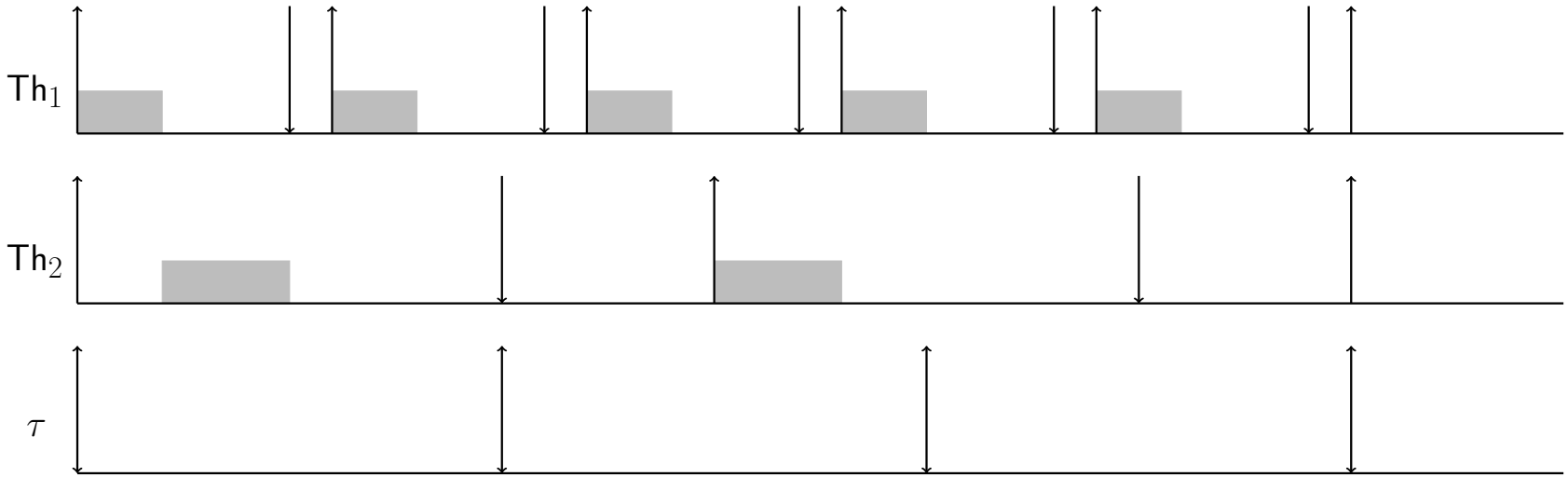




L'heuristique de partitionnement

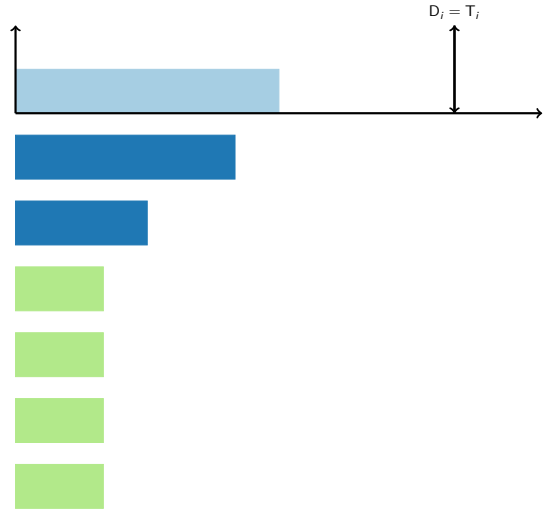
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

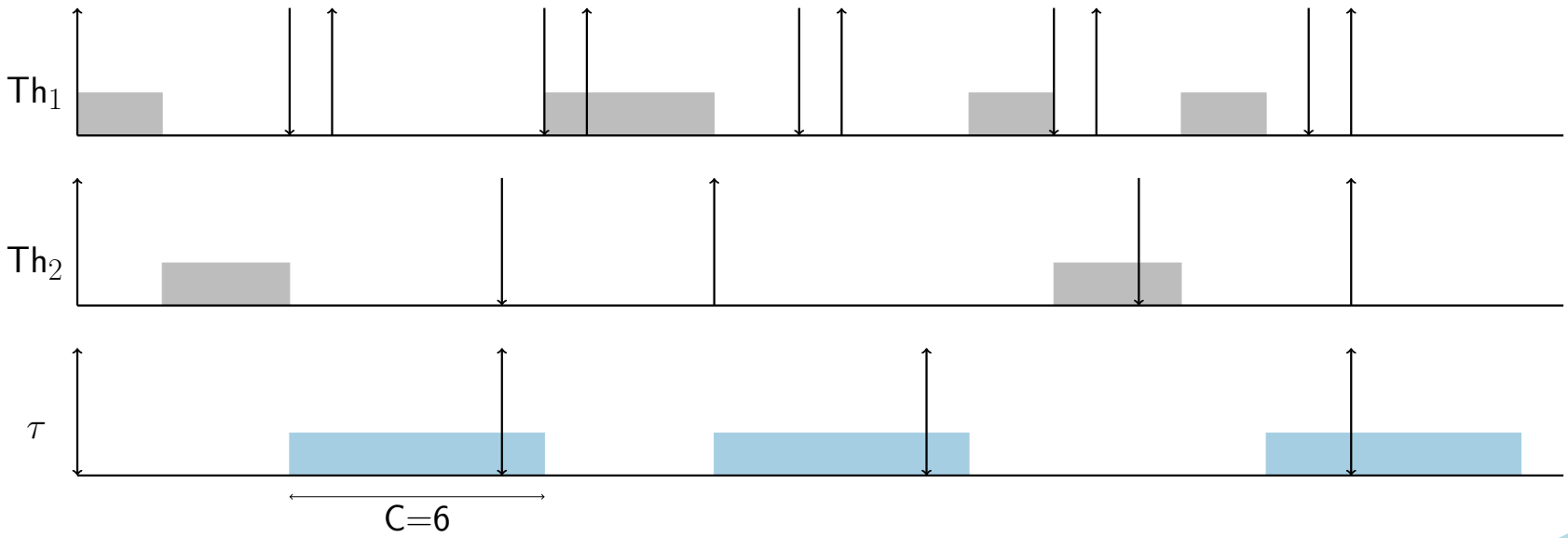




L'heuristique de partitionnement

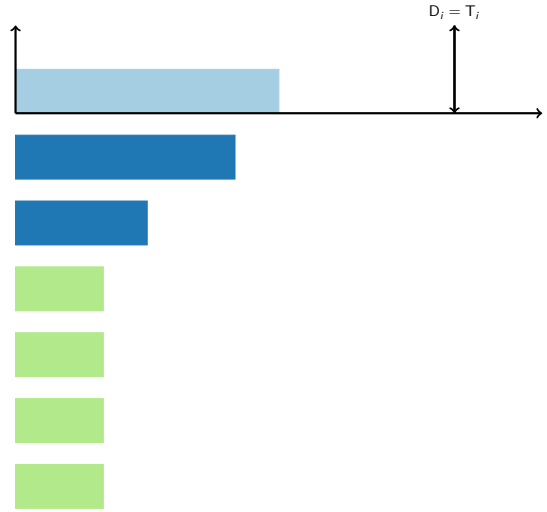
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

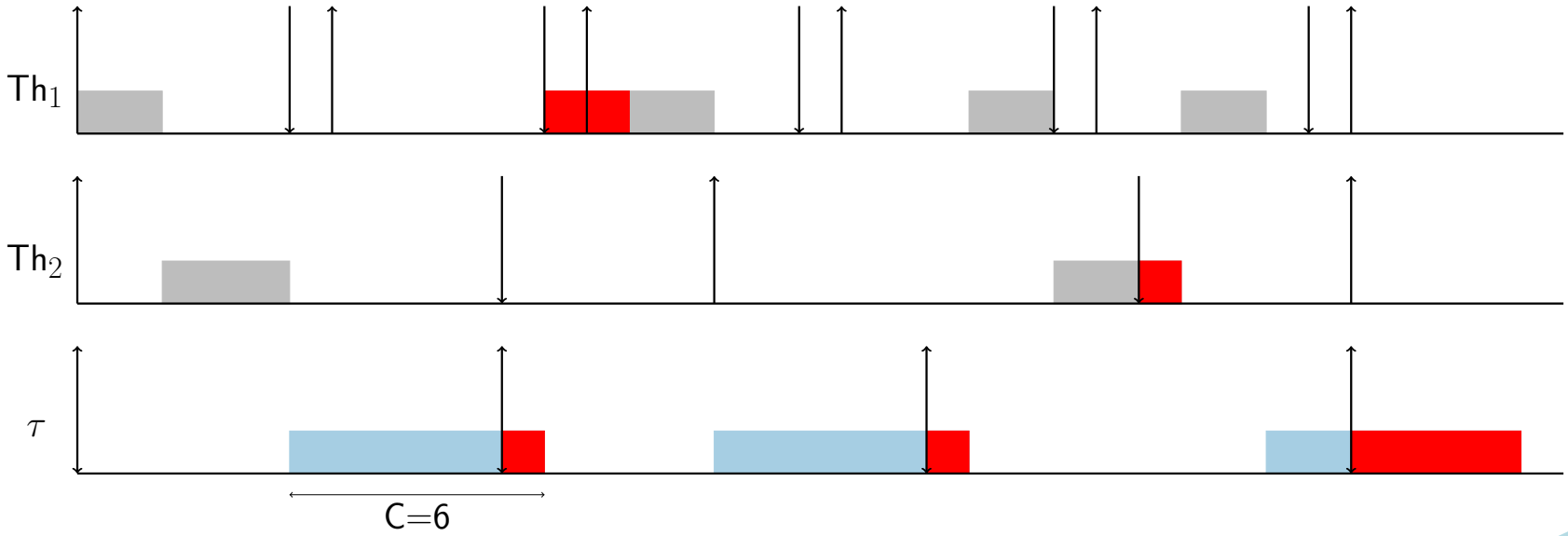




L'heuristique de partitionnement

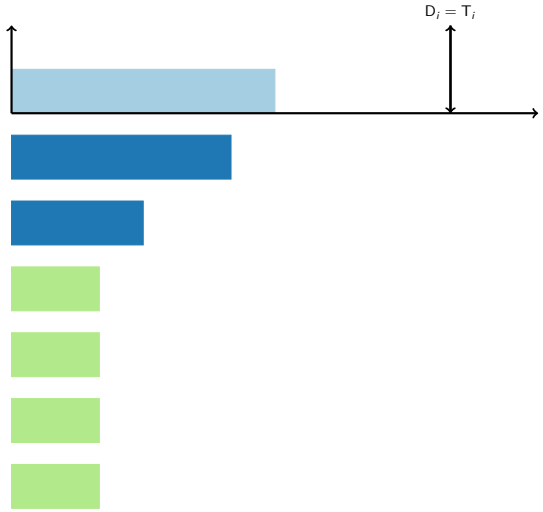
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

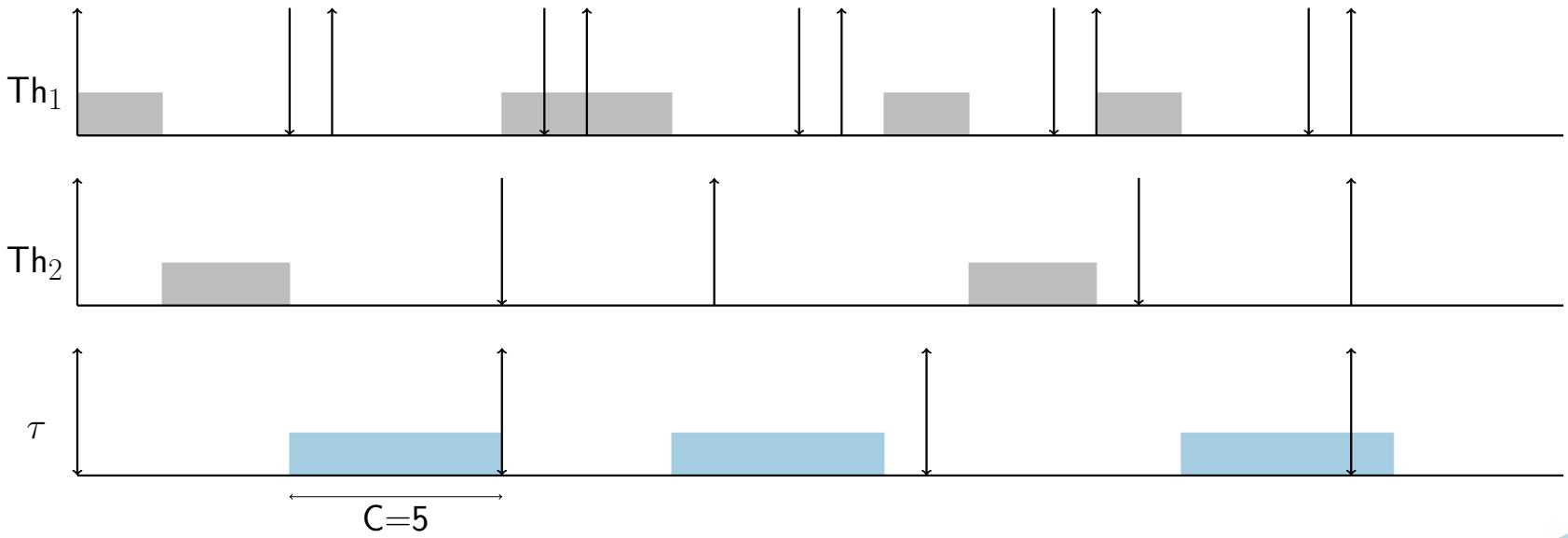




L'heuristique de partitionnement

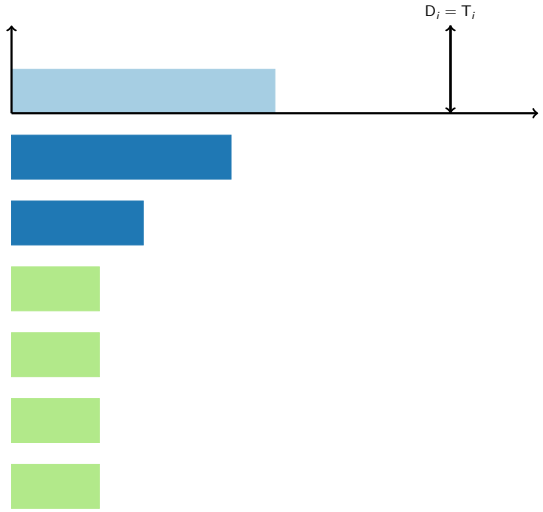
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

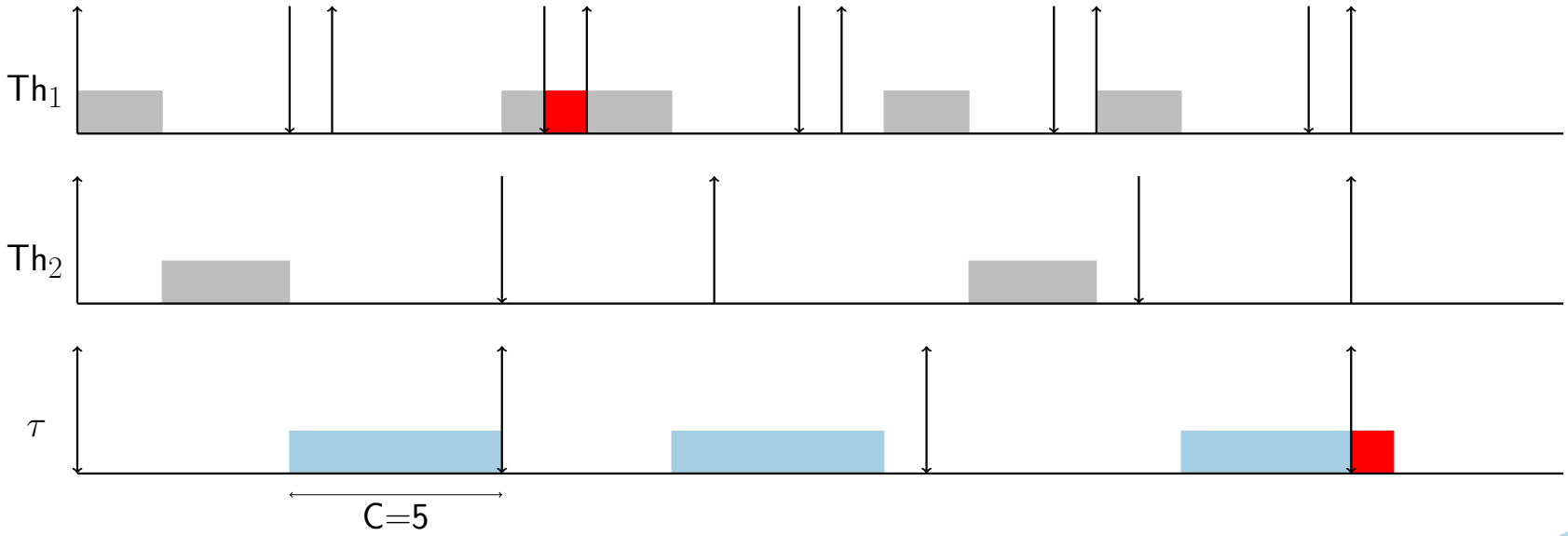




L'heuristique de partitionnement

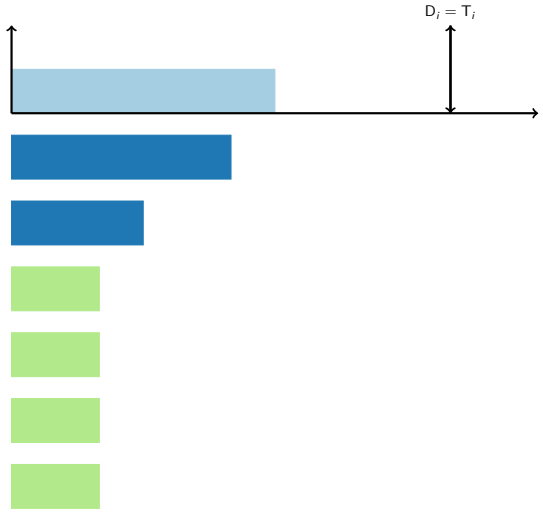
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

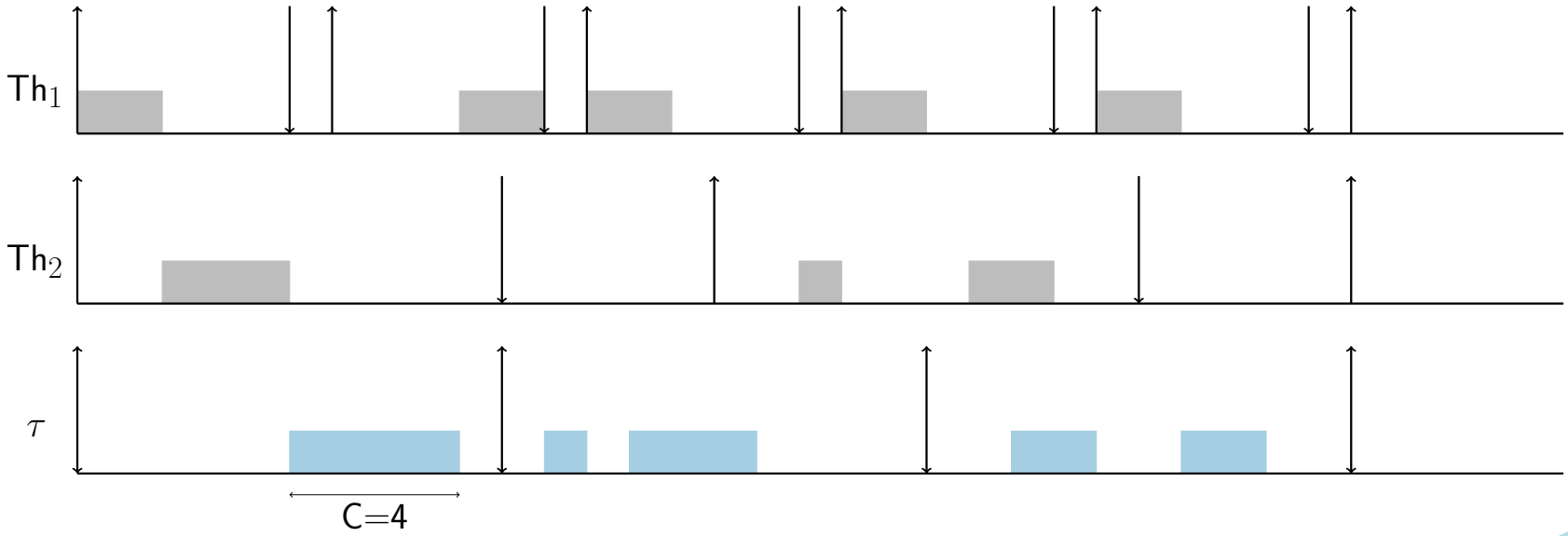




L'heuristique de partitionnement

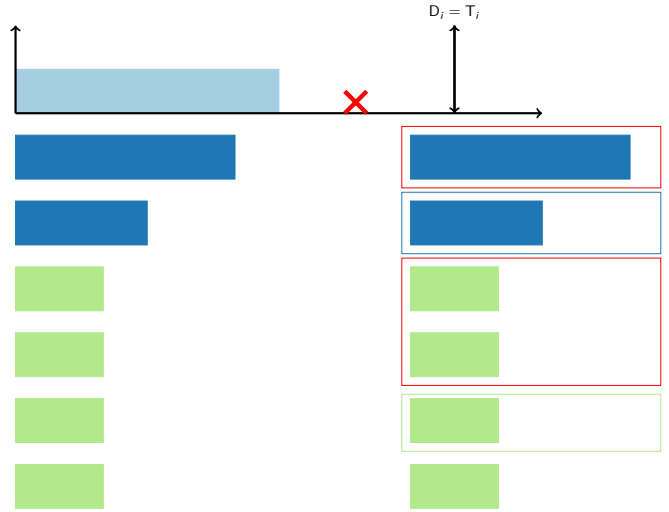
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

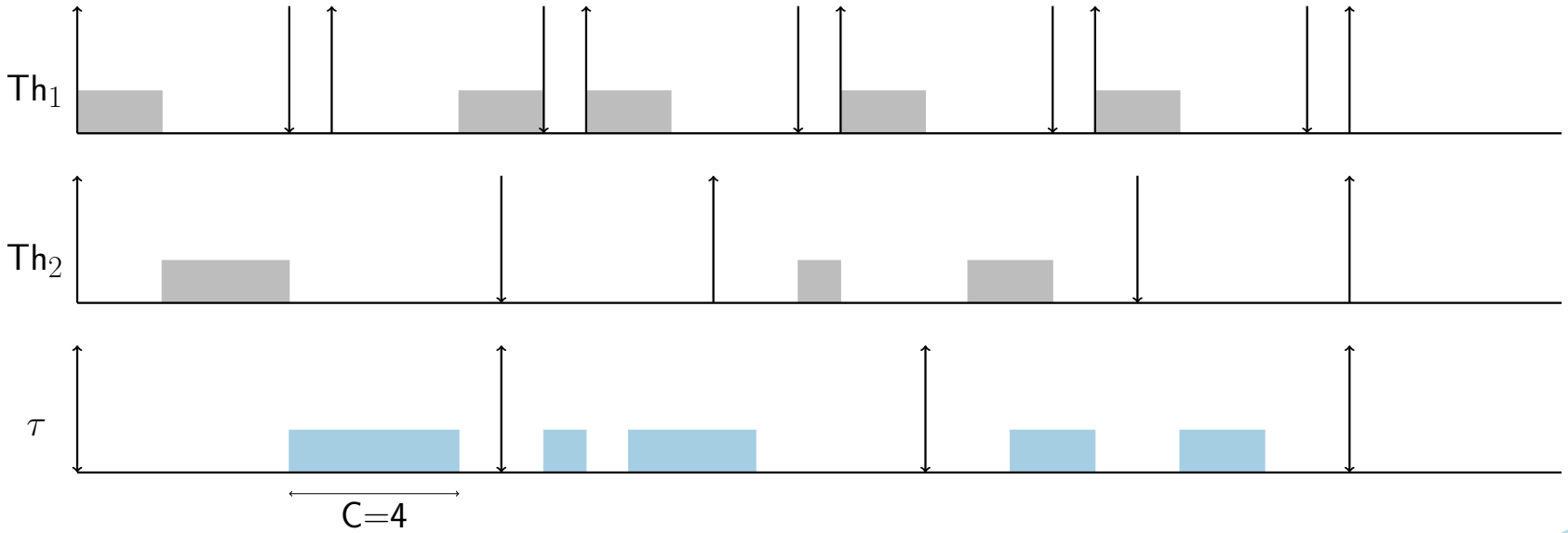




L'heuristique de partitionnement

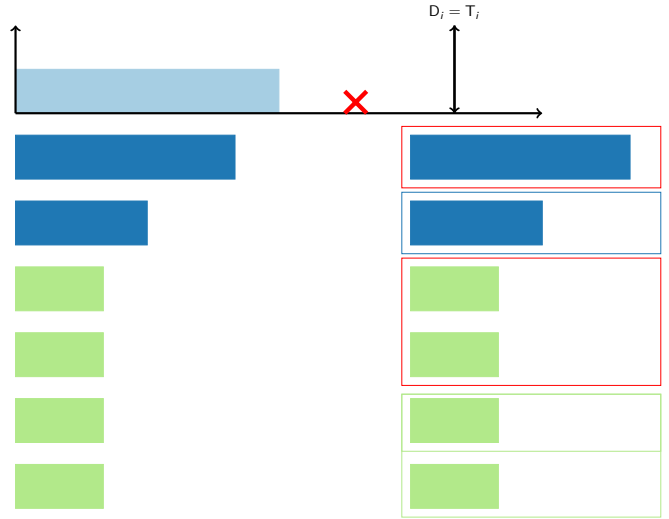
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

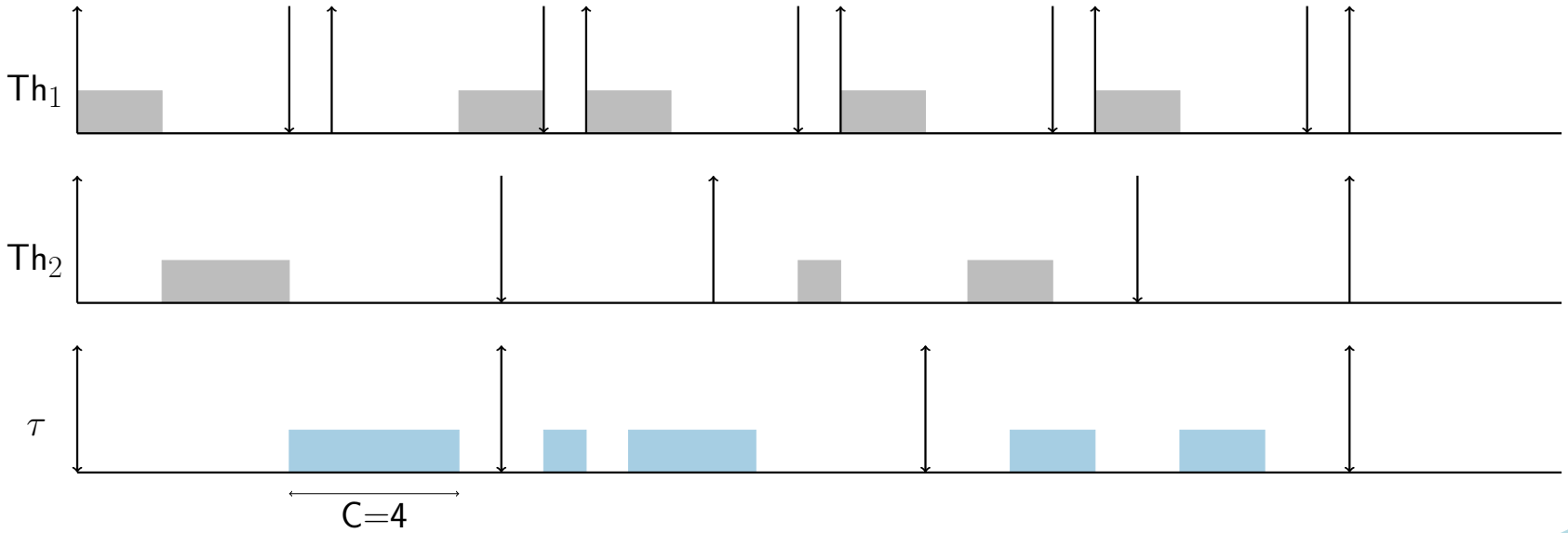




L'heuristique de partitionnement

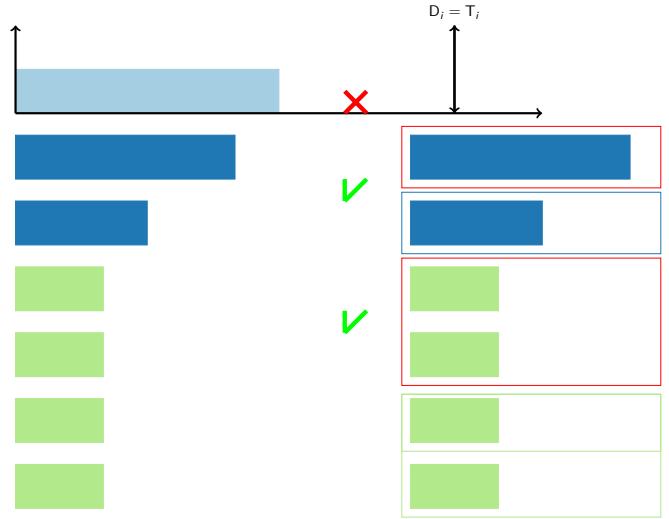
- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.



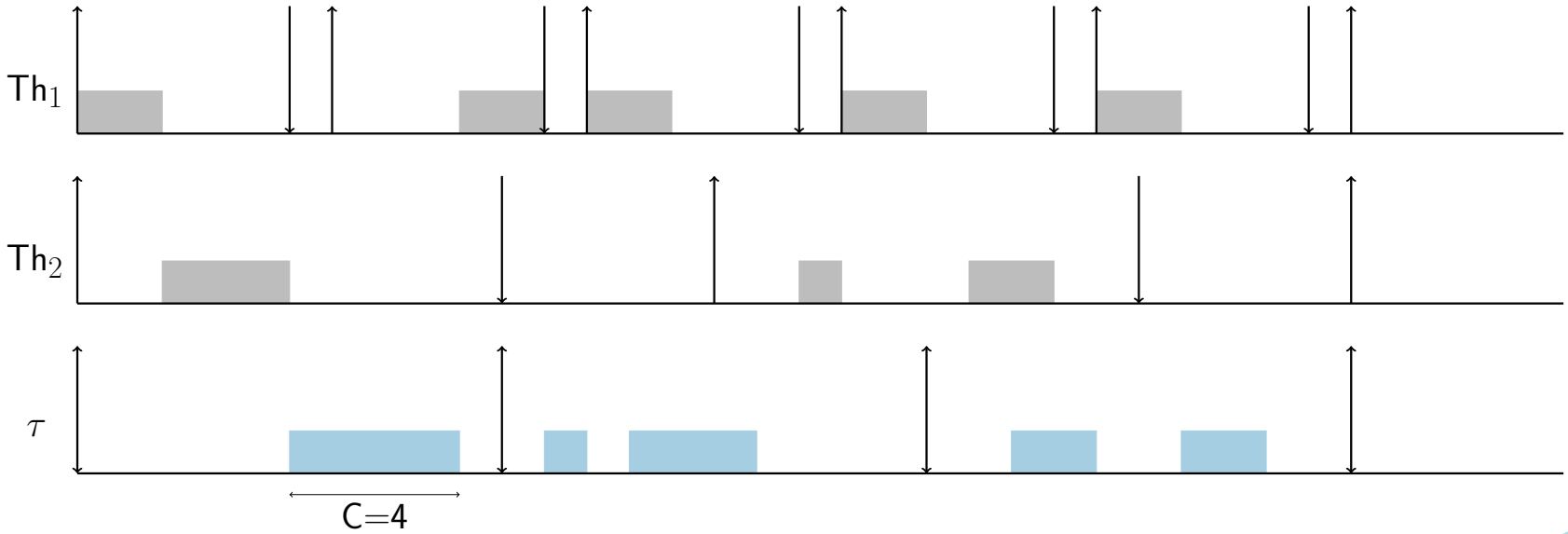


L'heuristique de partitionnement

- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : Th_1, Th_2 sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.

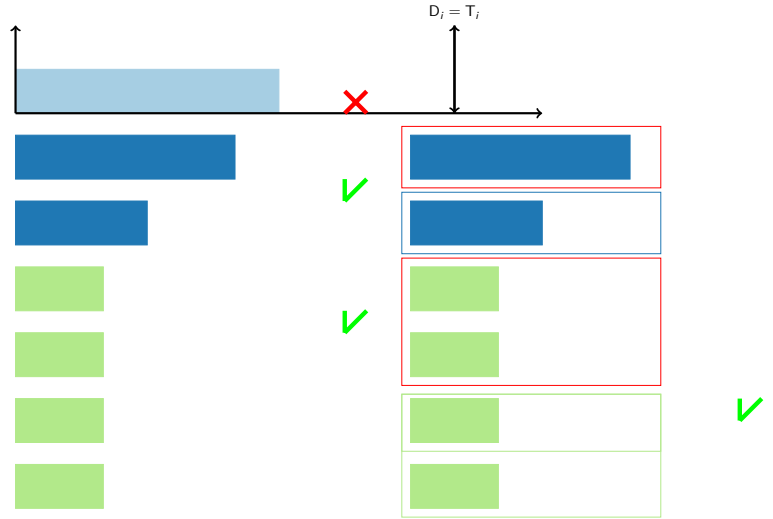


Threads maintenus d'un point de vue temps réel

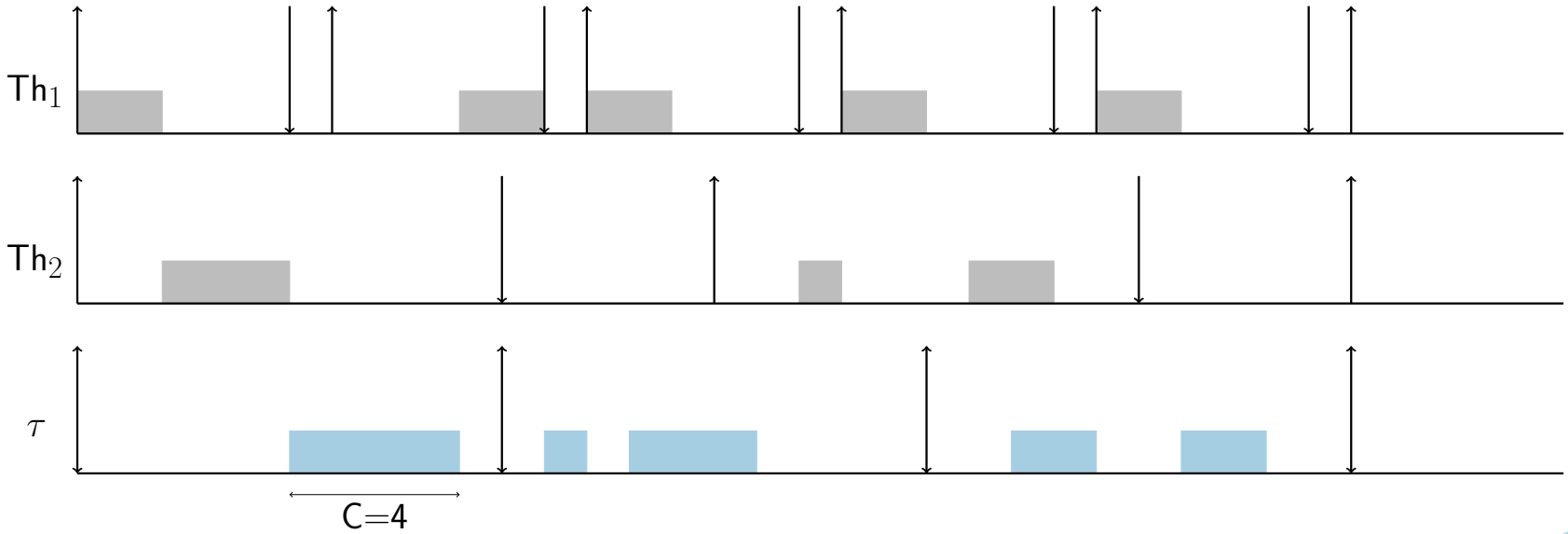


L'heuristique de partitionnement

- Définir le maximum de temps d'exécution d'une tâche qui peut être alloué à un processeur
- **Exemple** : $\mathrm{Th}_1, \mathrm{Th}_2$ sont deux threads alloués à un processeur et on essaye d'allouer τ (à droite) sur le même processeur.



Threads maintenus d'un point de vue temps réel



Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{\mathrm{f}_{\mathrm{op}}^{g}}{\mathrm{f}_{max}}\mathrm{nb}$

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f_{op}^{g}}{f_{max}}$nb

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}}$nb

Sélection de fréquence

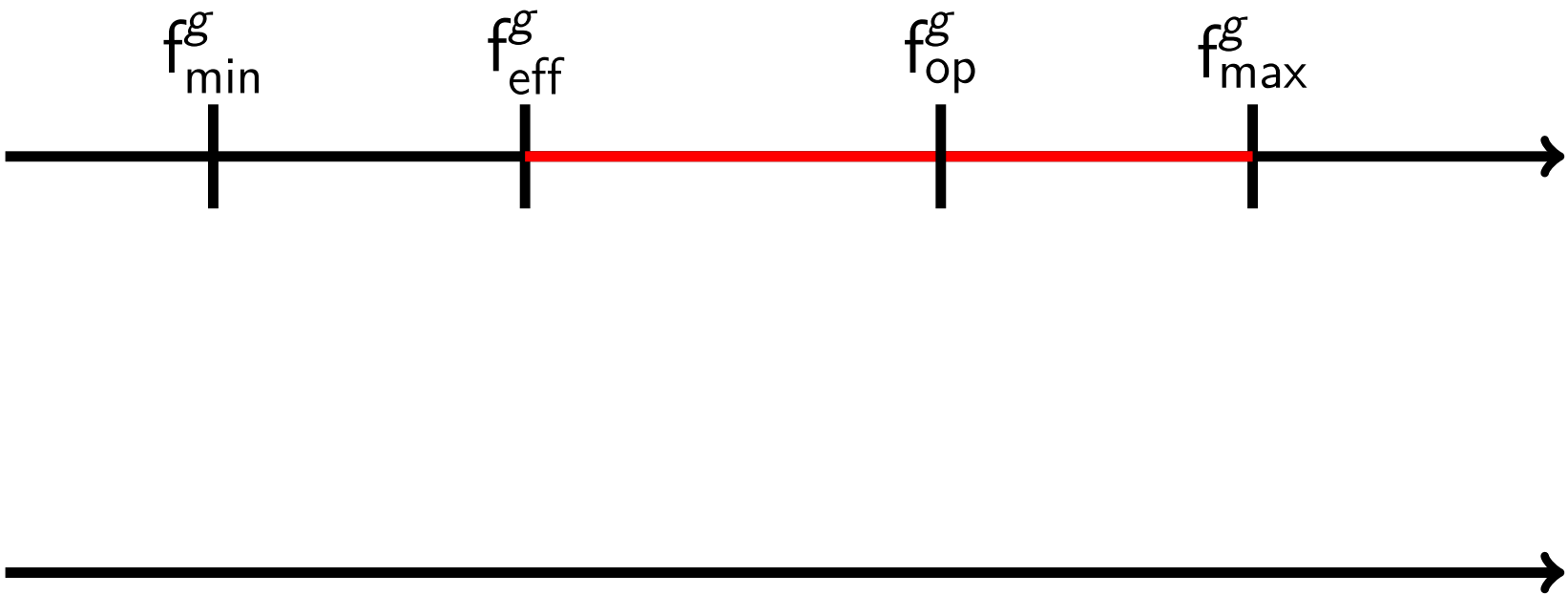
- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$



Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

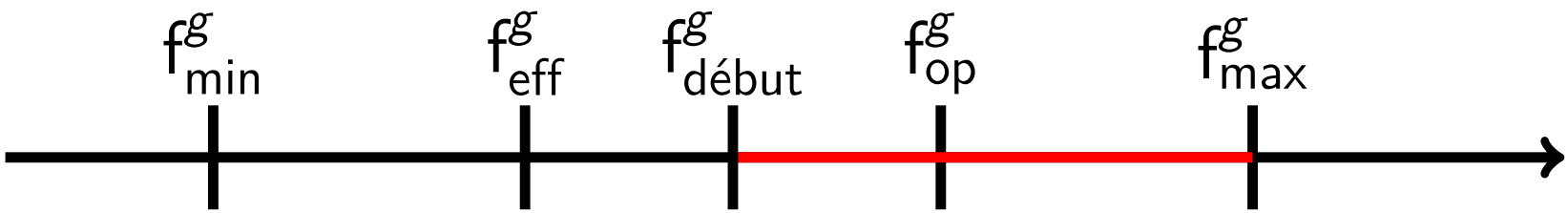
- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

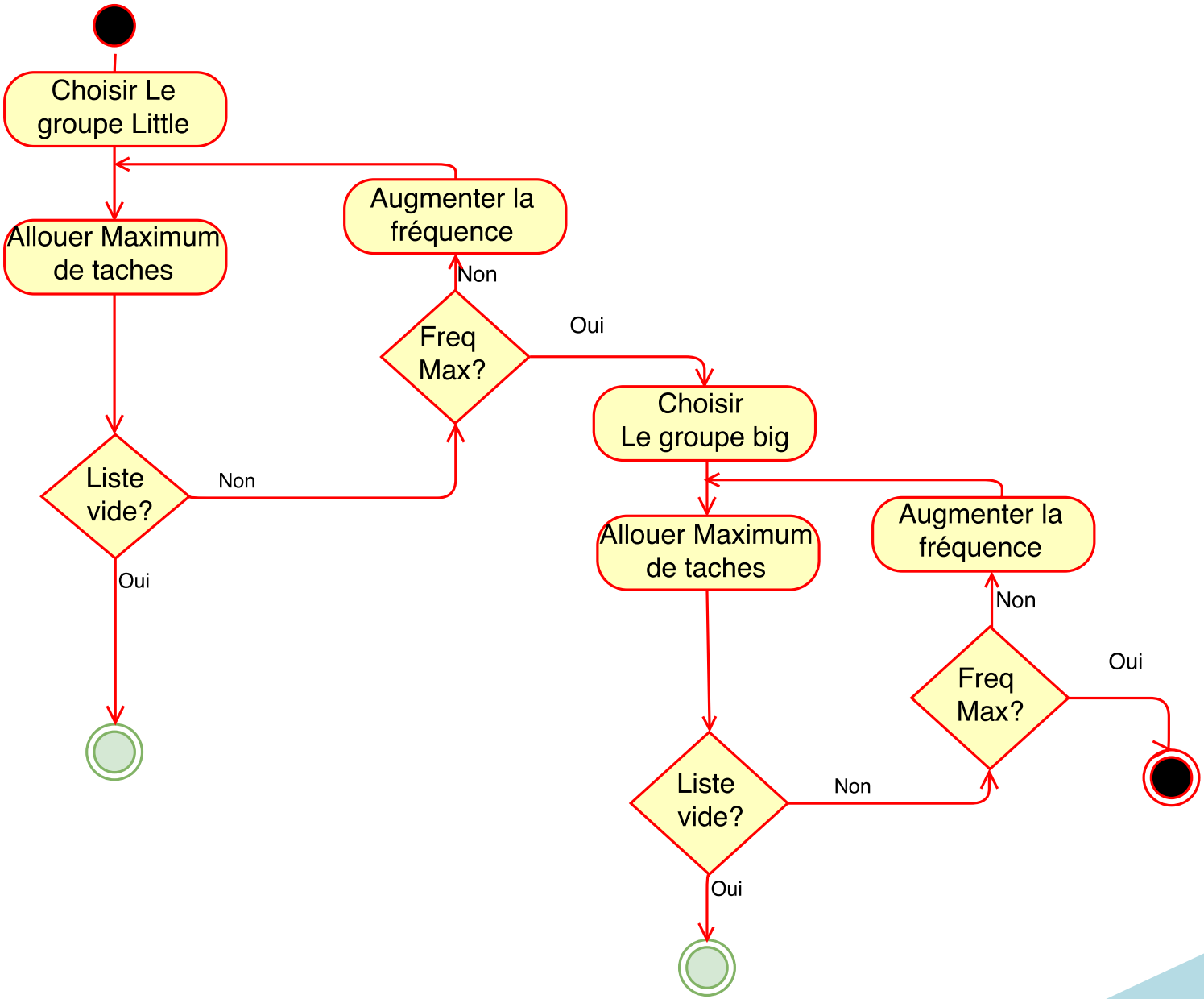
- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$

Sélection de fréquence

- La fréquence est entre la fréquence effective minimale et la fréquence max
- La puissance de calcul est définie comme $\frac{f^g_{op}}{f_{max}} nb$



Algorithme global



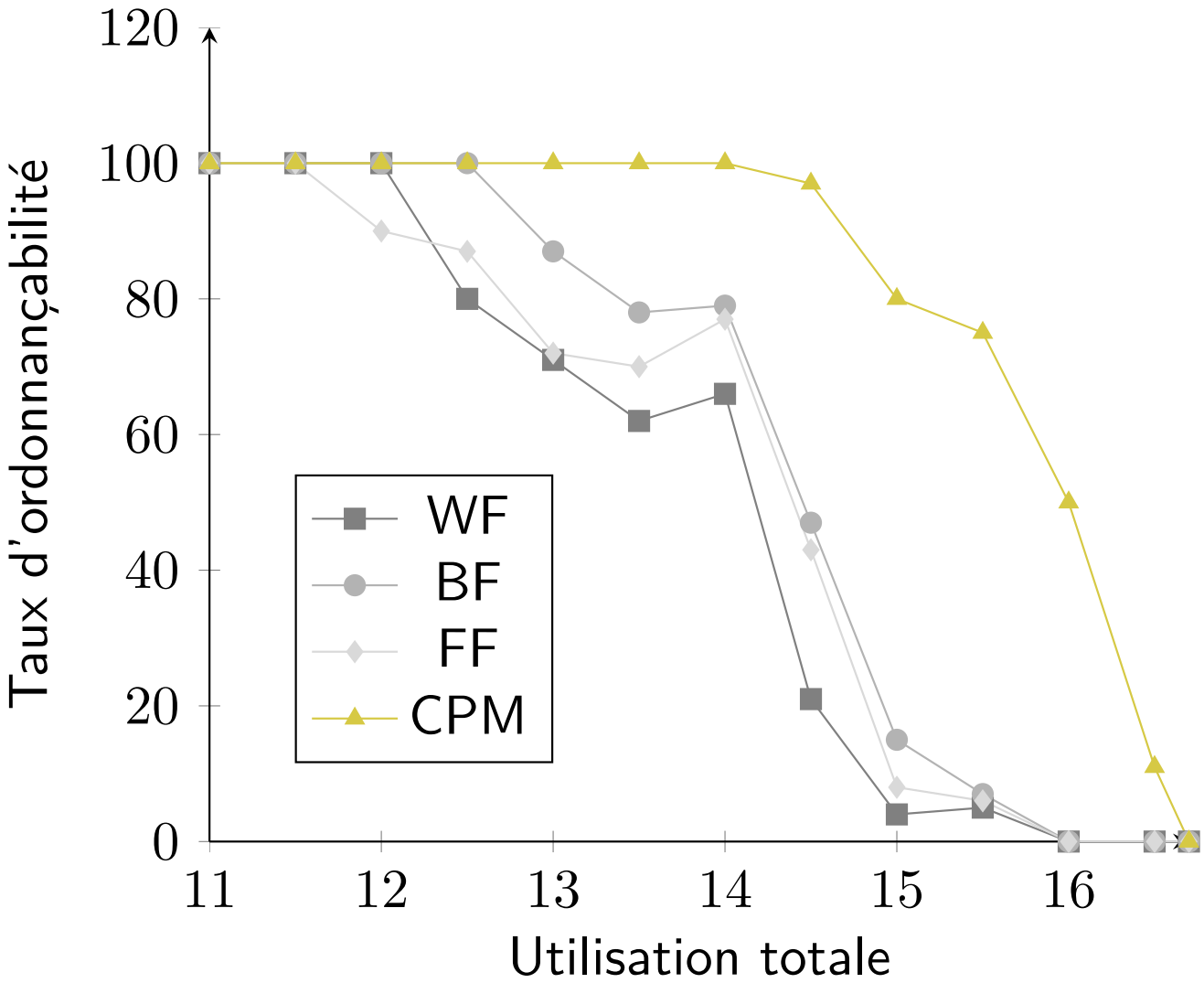
Protocole expérimental

- Plateforme hétérogène simulée : Exynos 5422.
- Utilisation générée (les coeurs littles comme référence) de 0.5 à 18.
- Comparer contre les heuristiques séquentielles Best Fit (BF), First Fit (FF), Worst Fit (WF)

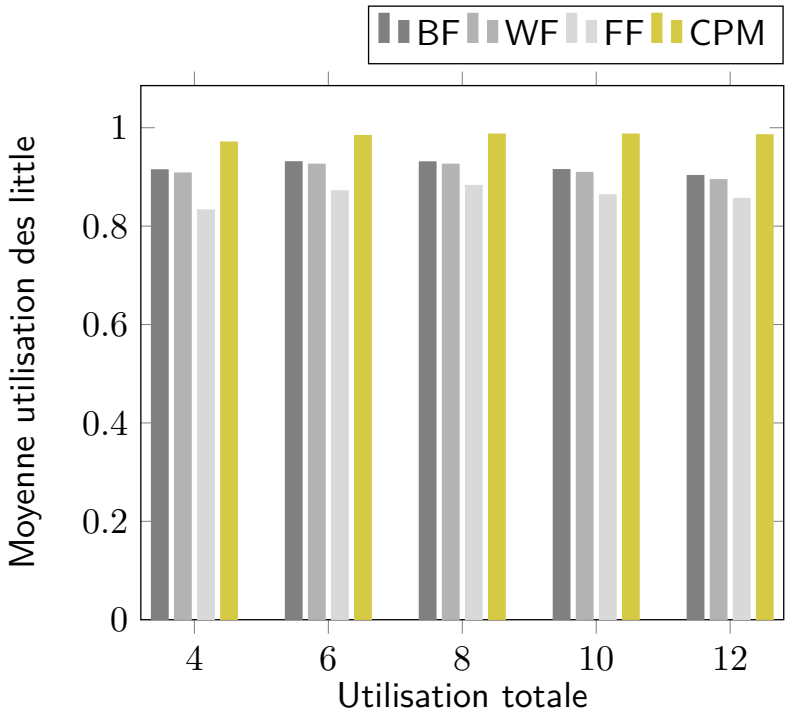
Scénarios :

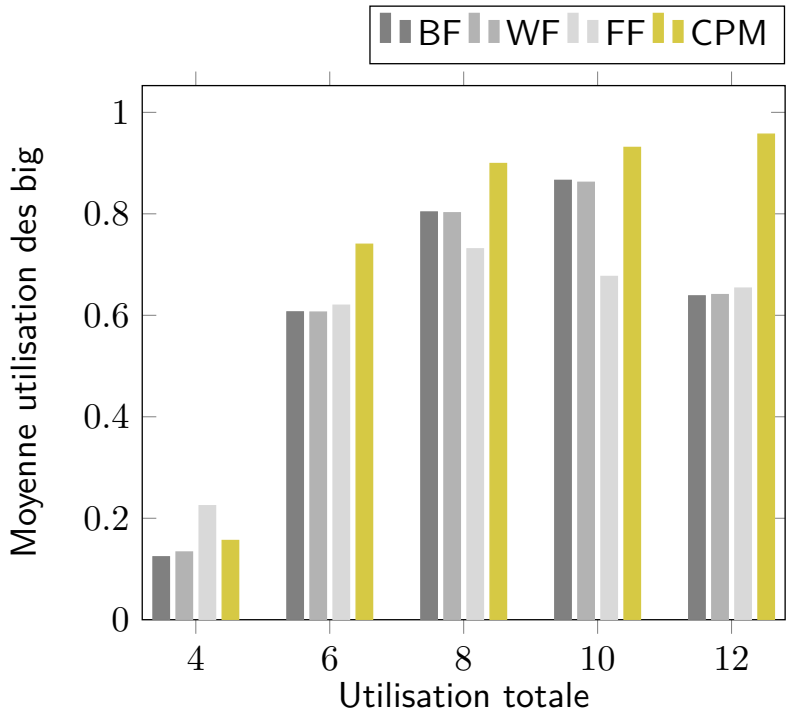
- Utilisation : $u_i = \frac{C_{i11}}{T_i} \leq 1$,
- Utilisation : $u_i = \frac{C_{i11}}{T_i}$ peut être > 1

Scénario 1

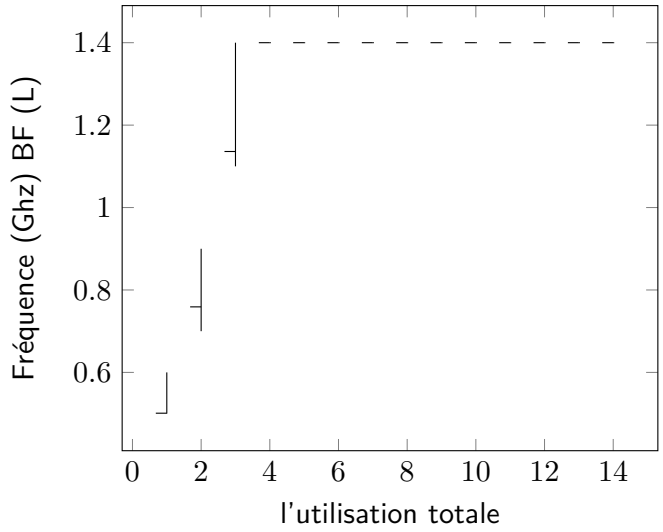

120
100
80
60
40
20
0
Taux d'ordonnançabilité
11
12
13
14
15
16
Utilisation totale
WF
BF
FF
CPM

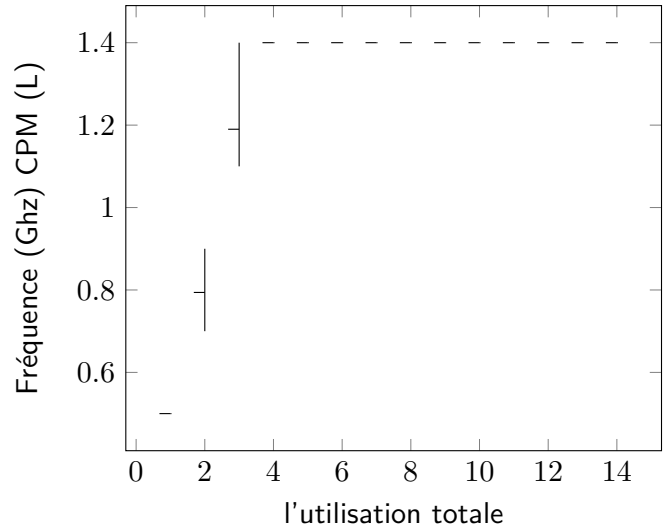
Scénario 1

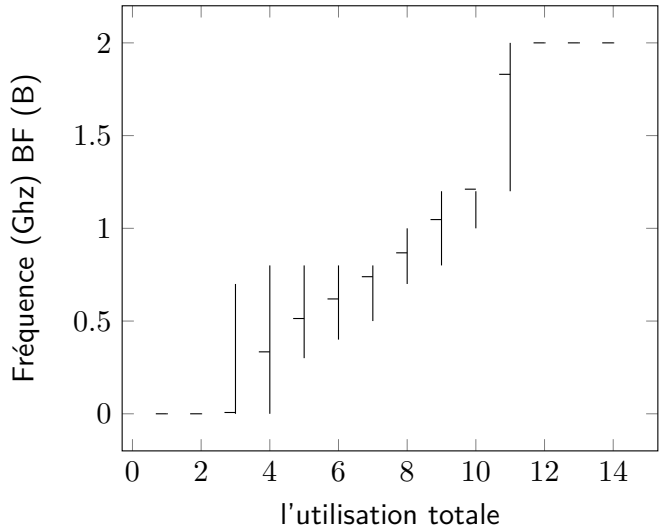

BF
WF
FF
CPM
Moyenne utilisation des little
0
0.2
0.4
0.6
0.8
1
4
6
8
10
12
Utilisation totale

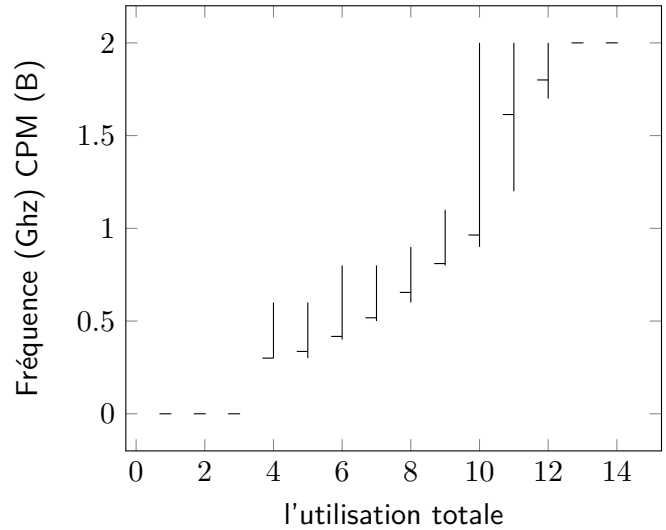

BF
WF
FF
CPM
Moyenne utilisation des big
0
0.2
0.4
0.6
0.8
1
4
6
8
10
12
Utilisation totale

Scénario 1

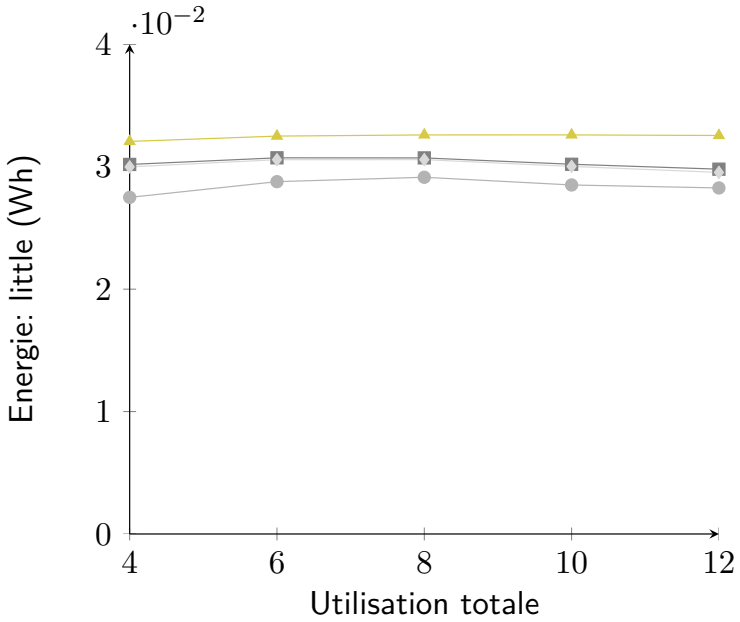

Fréquence (Ghz) BF (L)
1.4
1.2
1
0.8
0.6
0
2
4
6
8
10
12
14
l'utilisation totale

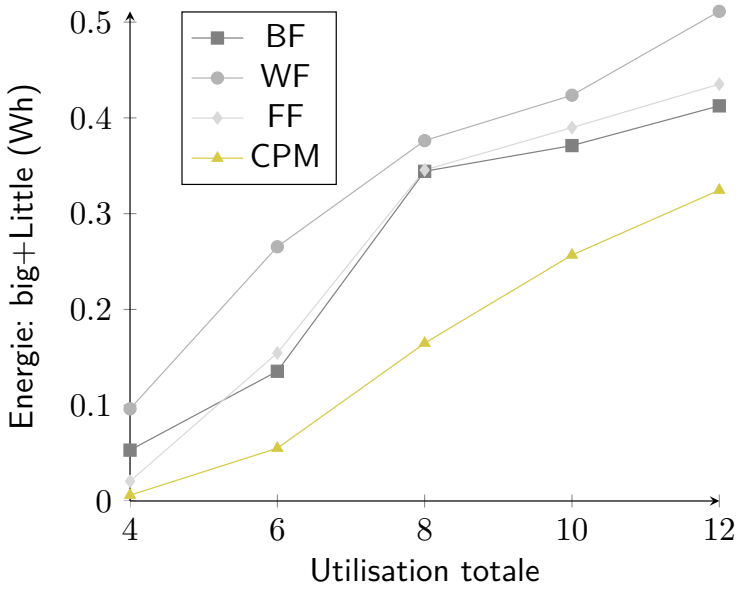

Fréquence (Ghz) CPM (L)
1.4
1.2
1
0.8
0.6
0
2
4
6
8
10
12
14
l'utilisation totale


Fréquence (Ghz) BF (B)
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
l'utilisation totale


Fréquence (Ghz) CPM (B)
2
1.5
1
0.5
0
0
2
4
6
8
10
12
14
l'utilisation totale

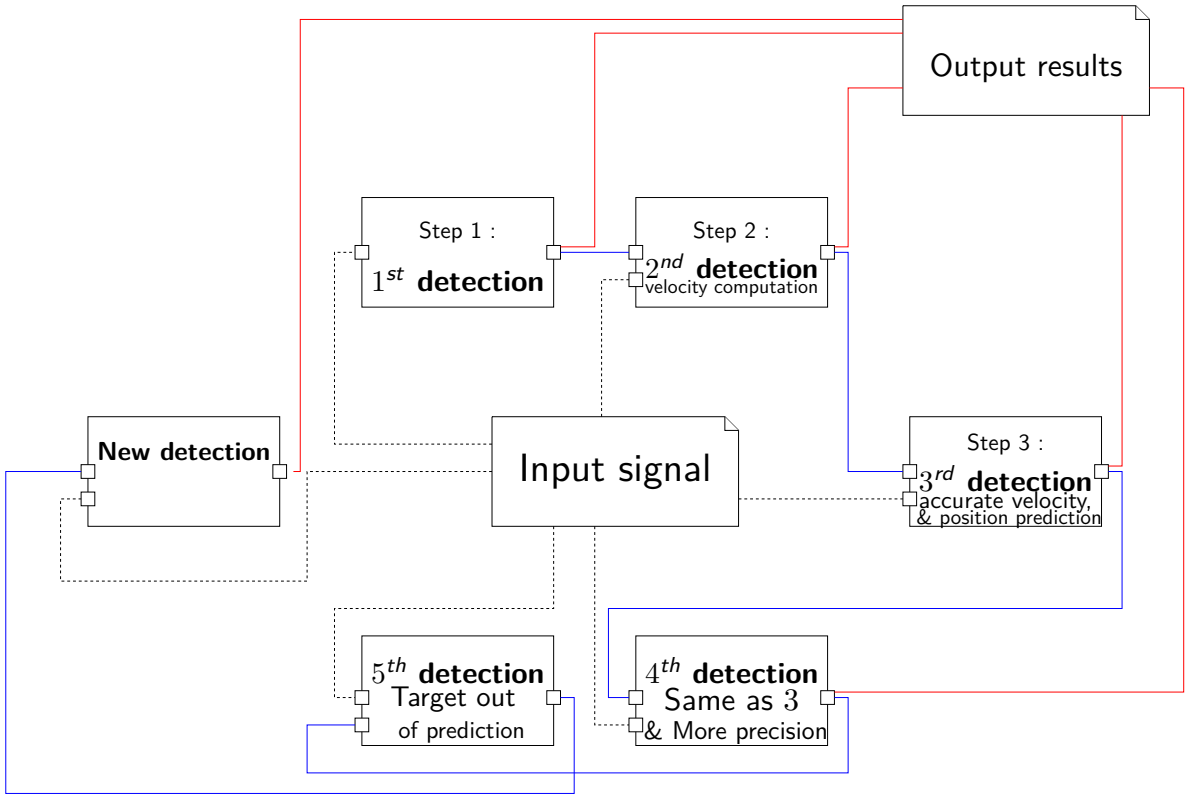
Scénario 1


$\cdot 10^{-2}$
4
3
2
1
0
Energie: little (Wh)
4
6
8
10
12
Utilisation totale


0.5
0.4
0.3
0.2
0.1
0
Energie: big+Little (Wh)
BF
WF
FF
CPM
4
6
8
10
12
Utilisation totale

Plan

Radar MTI



- Le traitement dépend de la valeur des données en entrée
- Le traitement à faire est parallélisable

Une tâche parallèle avec digraphe

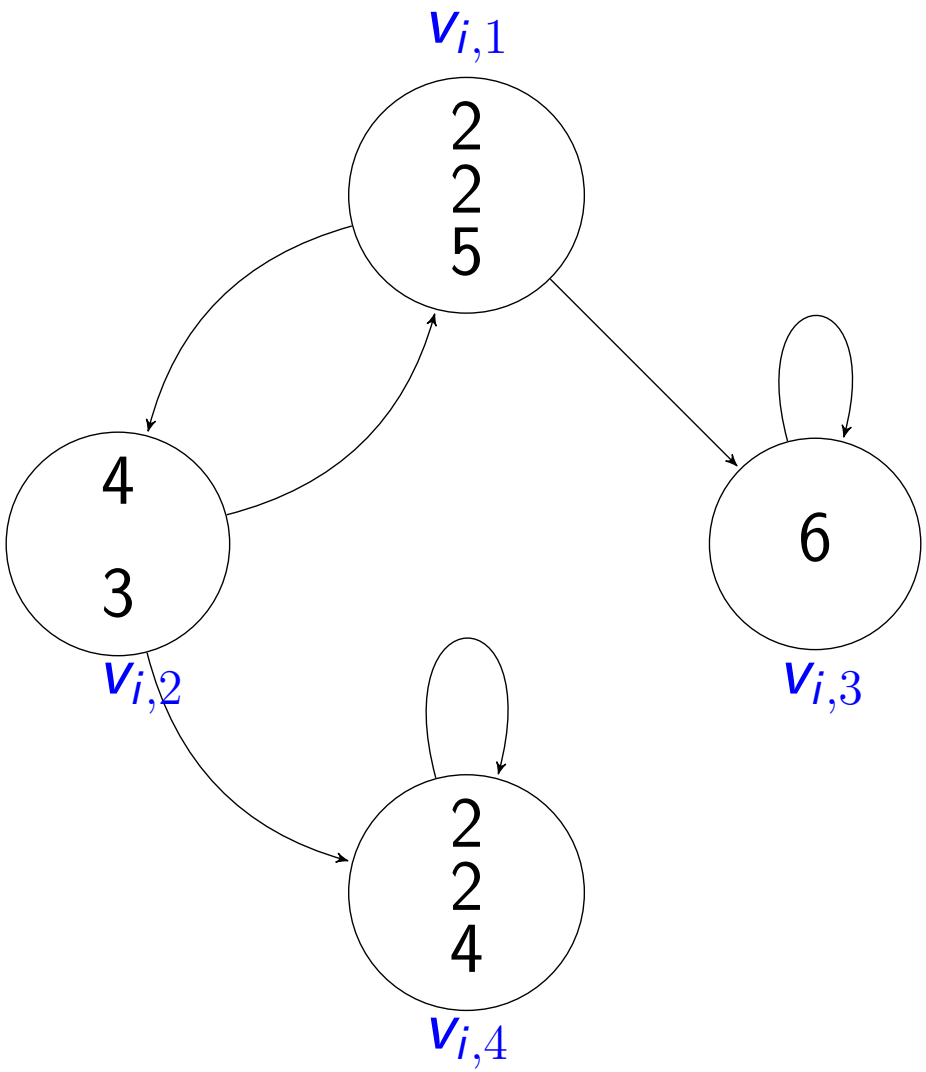


FIGURE : τ_i digraphe, $\mathsf{D}_i = 10, \mathsf{T}_i = 12$

Une tâche est :

- Sporadique (échéance contrainte $\mathsf{D}_i \leq \mathsf{T}_i$)
- Modélisée par un *automate*

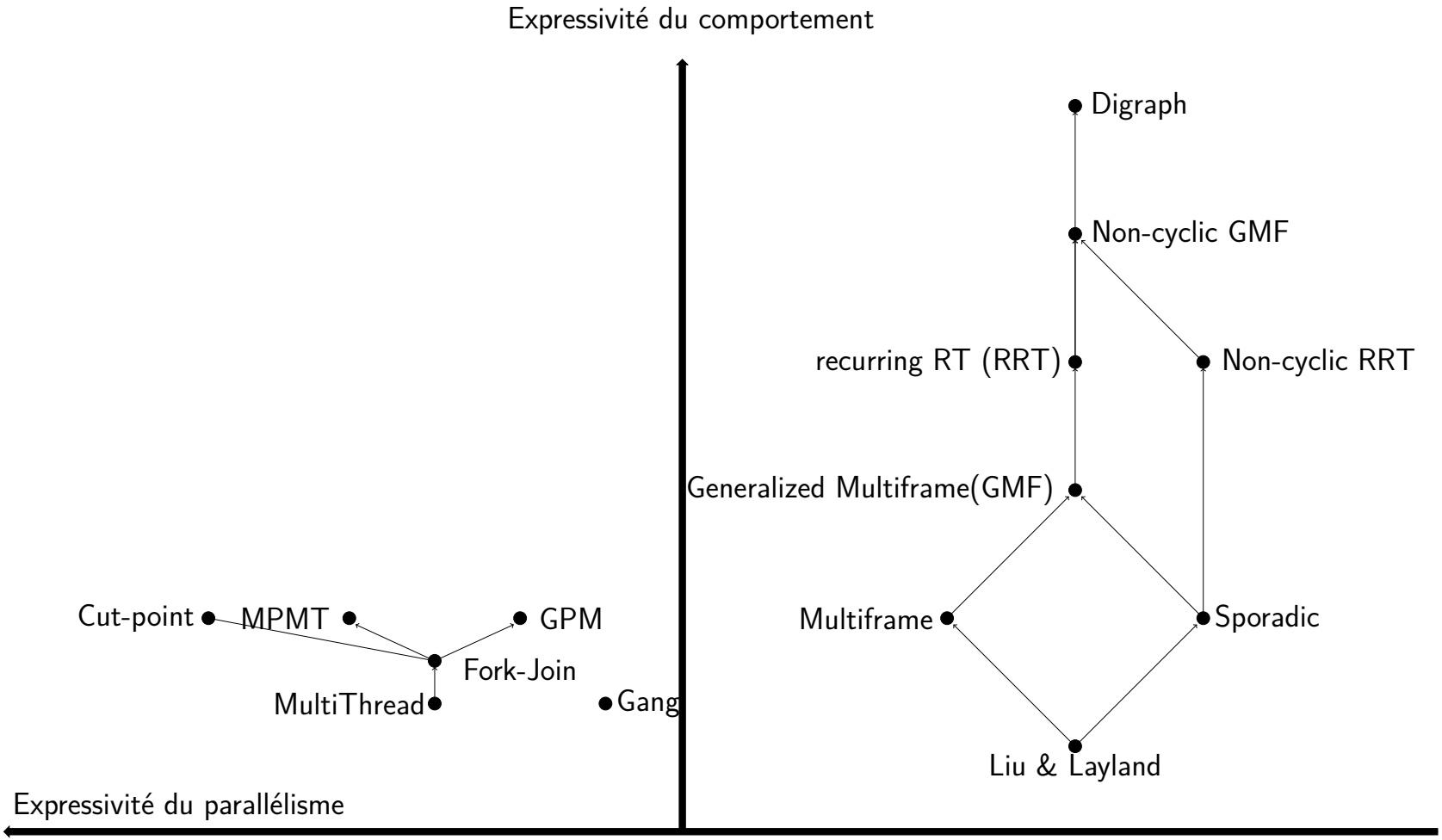
Chaque état :

- est une instance de la tâche
- composée de threads.

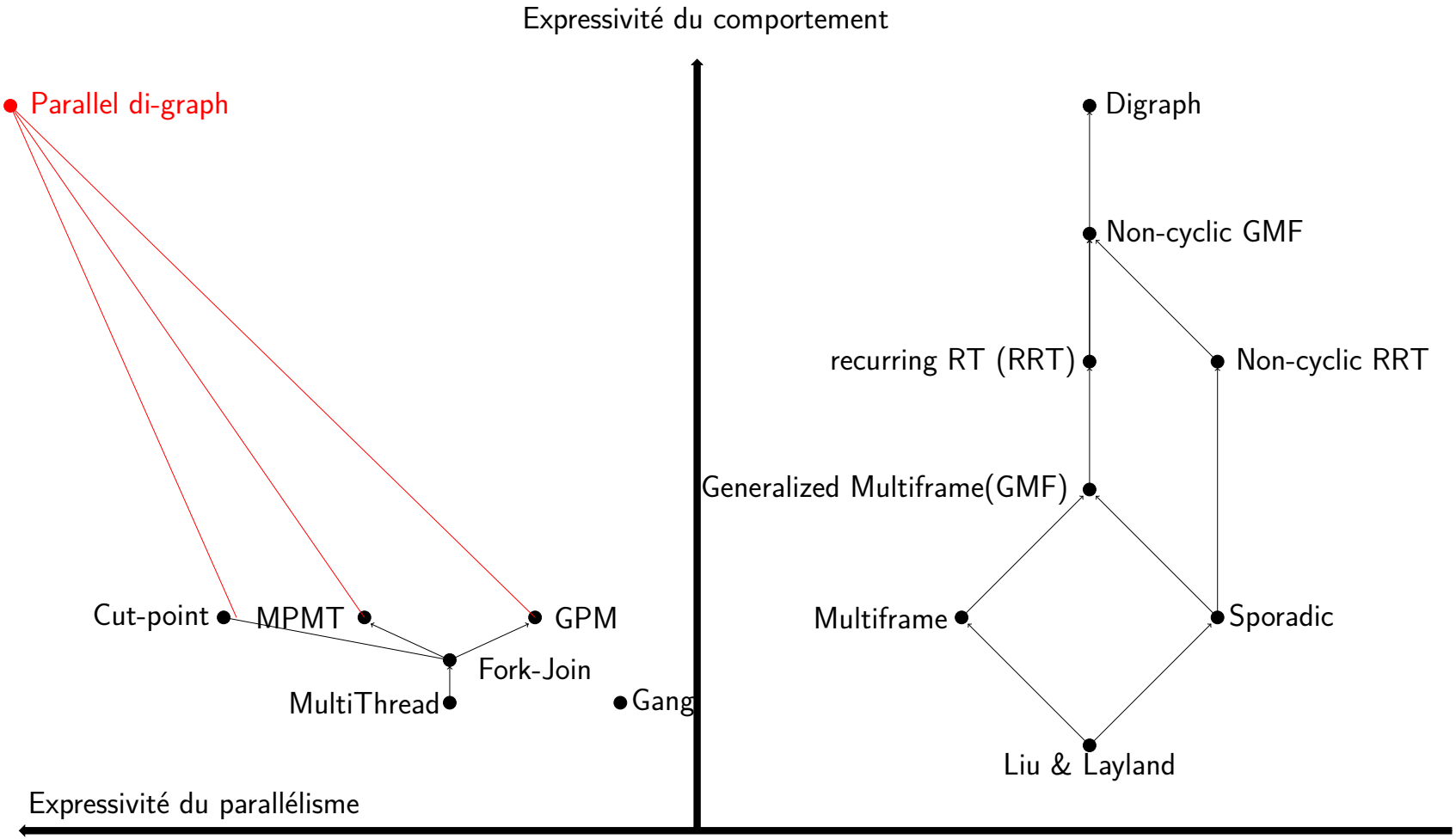
La transition entre états :

- Précédence entre deux instances.
- Non-déterministe

Les modèles de tâches



Les modèles de tâches



Formulation du problème

Nous avons :

- Un ensemble de tâches digraphes
- Un ensemble de coeurs identiques qui opèrent sur la même fréquence variable.

Objectif

- Allouer les digraphes sur les coeurs
- Sélectionner la fréquence minimale

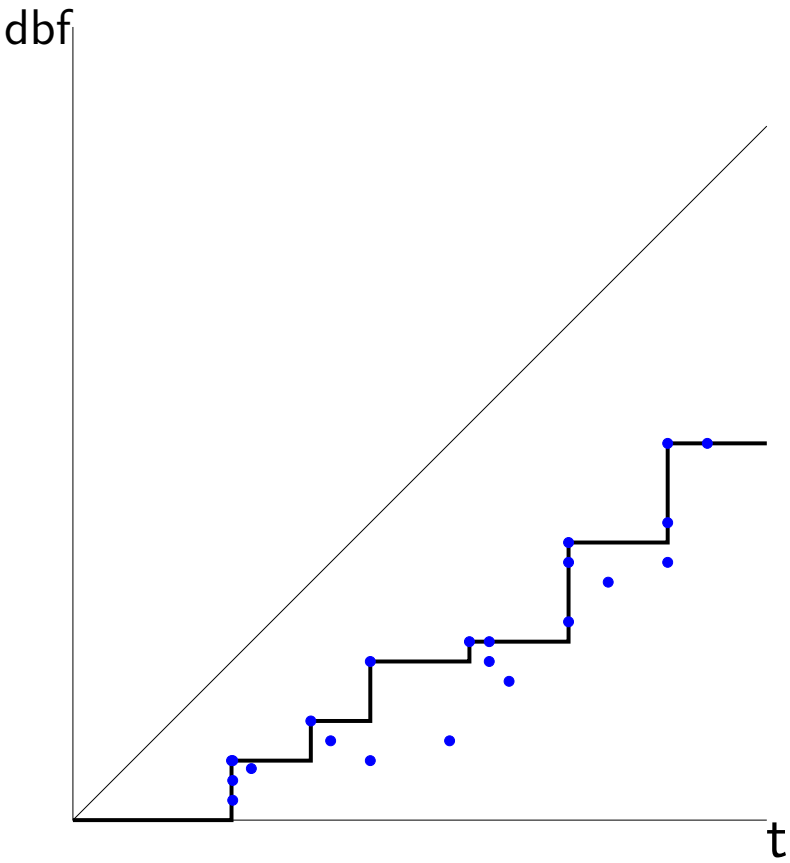
Contraintes

- Toutes les échéances doivent être respectées

Uniprocesseur

Uniprocesseur

Soit $\mathcal{T}$ un ensemble de tâches, et t un entier positif. $\mathcal{T}$ est ordonnançable sous EDF si et seulement si : (Stigge et al.). $\forall t, \text{dbf}(\mathcal{T}, t) \leq t$



Pour calculer la dbf$(\mathcal{T}, t)$:

1. Calculer la *df* pour tous les chemins, pour chaque tâche $\tau_i \in \mathcal{T}$
2. Pour chaque tâche et chemin, choisir la maximum *df* pour toutes les valeurs de t
3. Additionner les résultats du 2.

Partitionnement des digraphes

- EDF-partitionné.

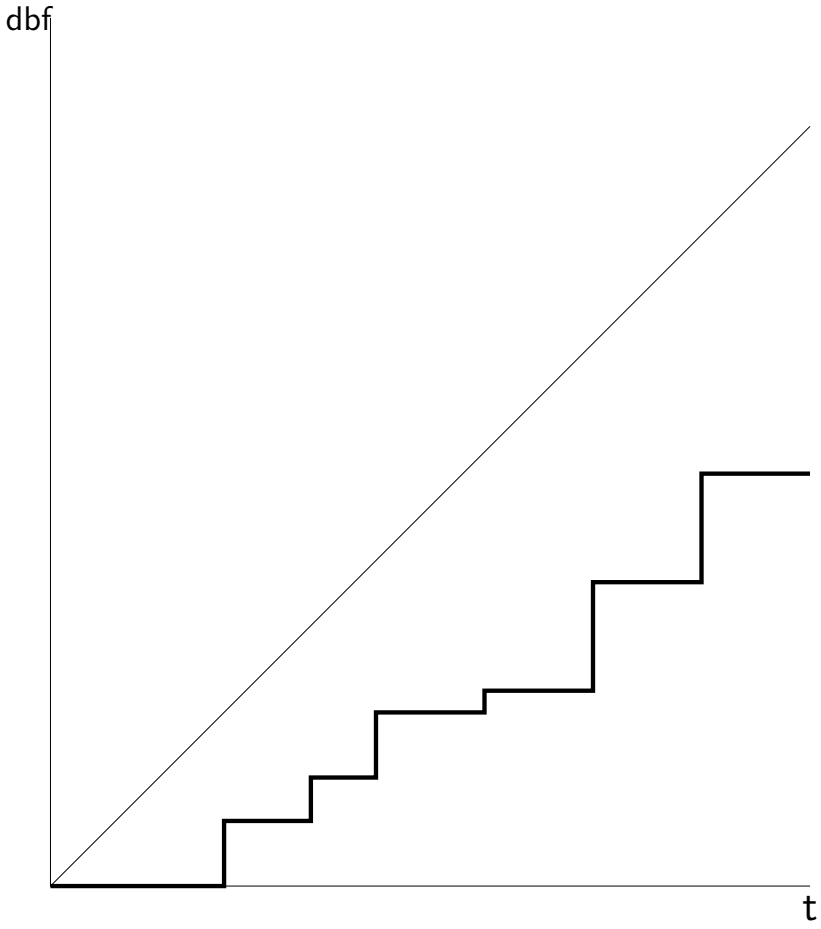
Solution Exacte

Vérifier la combinaison de toutes les tâches, états, threads, sur tous les coeurs et toutes les fréquences.

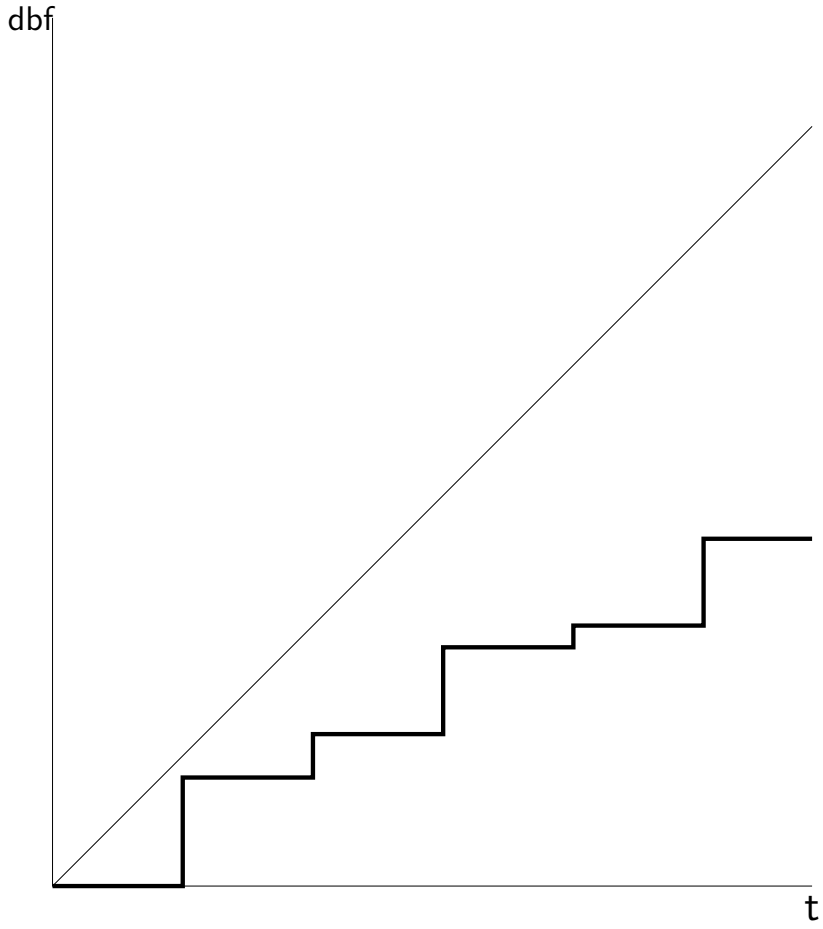
Pour chaque processeur et chaque tâche :

1. Essayer d'allouer La tâche en séquentiel.
2. Si non, elle est décomposée en deux parties :
   - La première partie est allouée au coeur courant
   - La deuxième est remise à la liste de tâches non allouées.

Partitionnement :

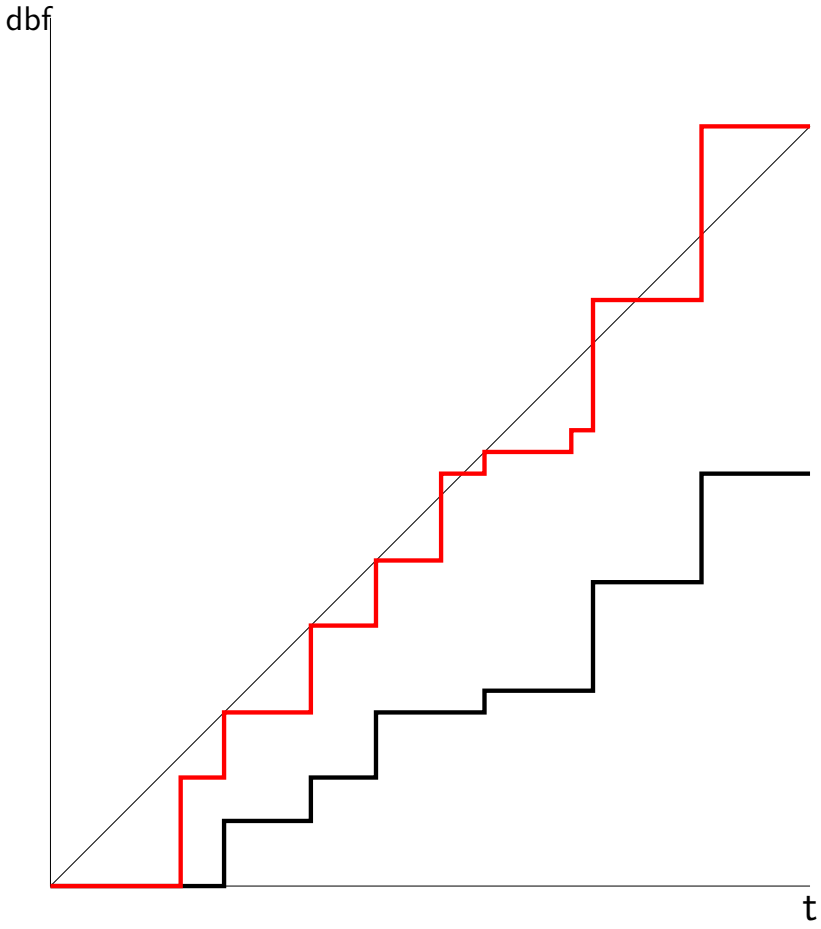


L'ensemble de tâches $\mathcal{T}$

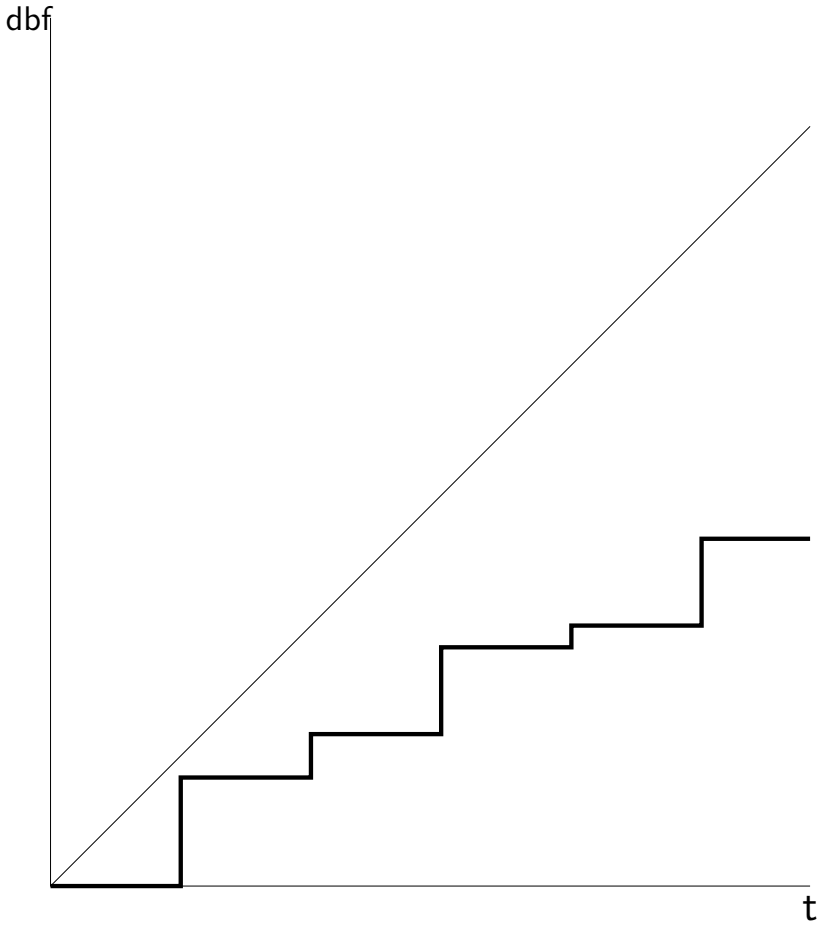


La tâche τ

Partitionnement :

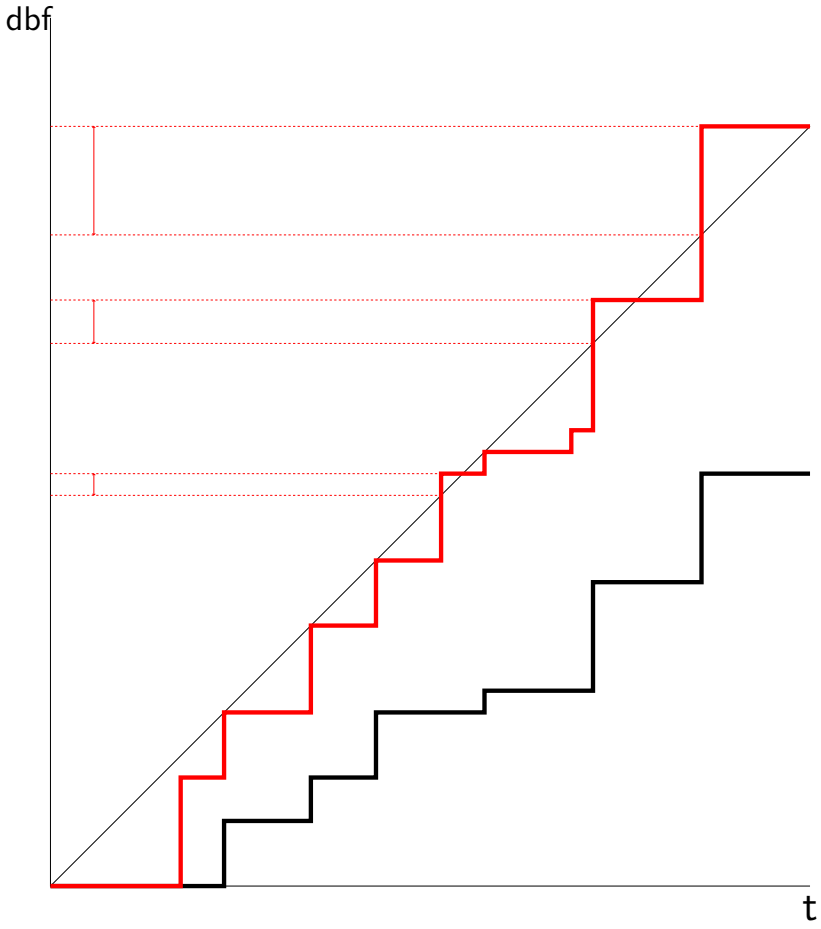


L'ensemble de tâches $\mathcal{T}$

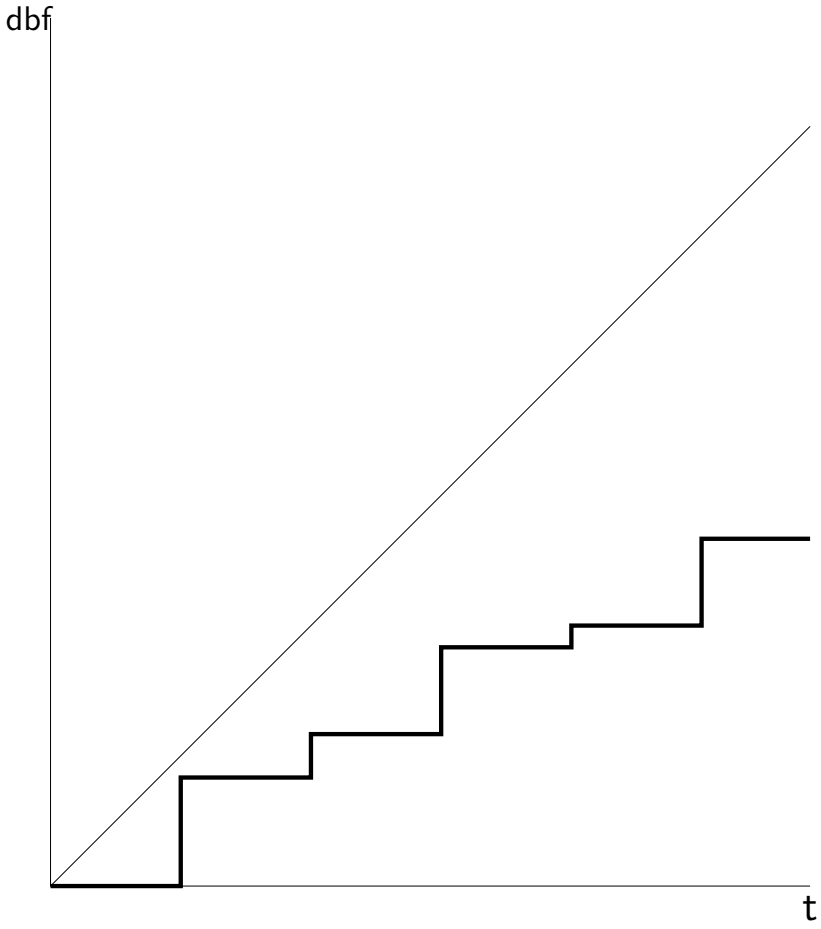


La tâche τ

Partitionnement :

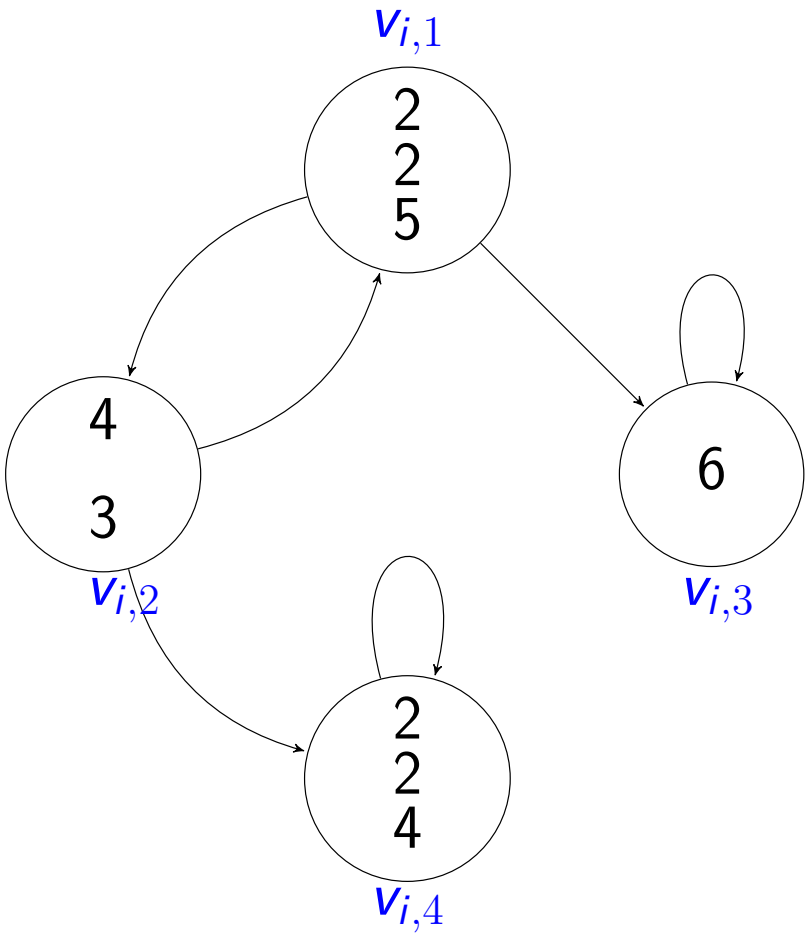


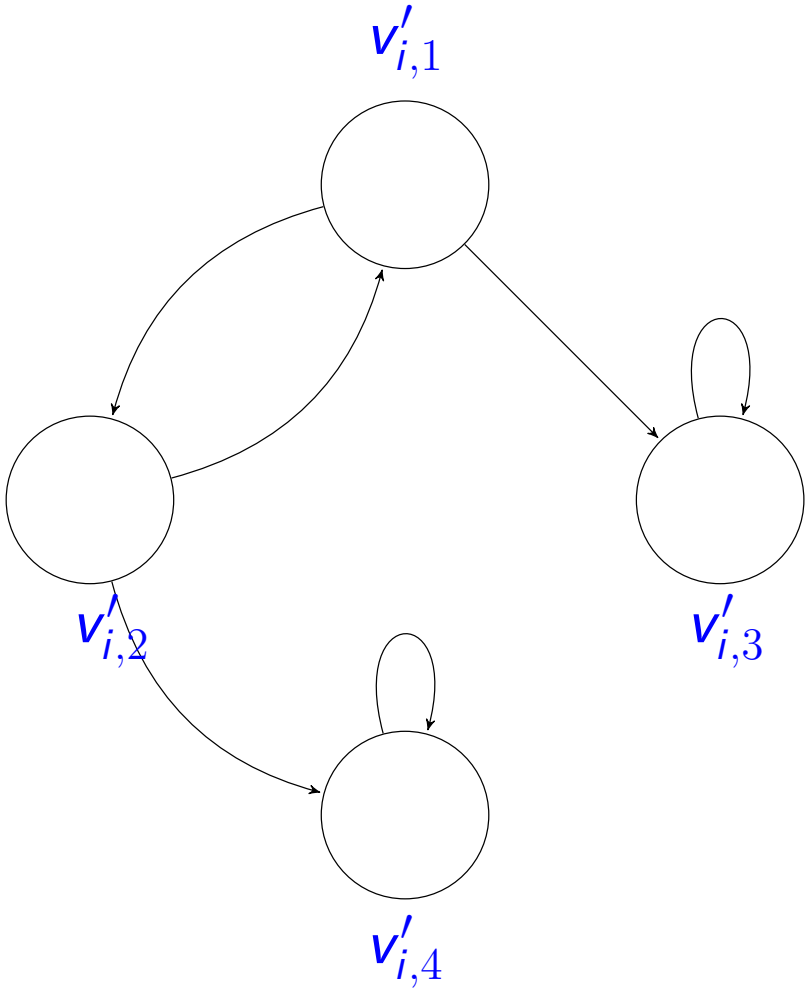
L'ensemble de tâches $\mathcal{T}$



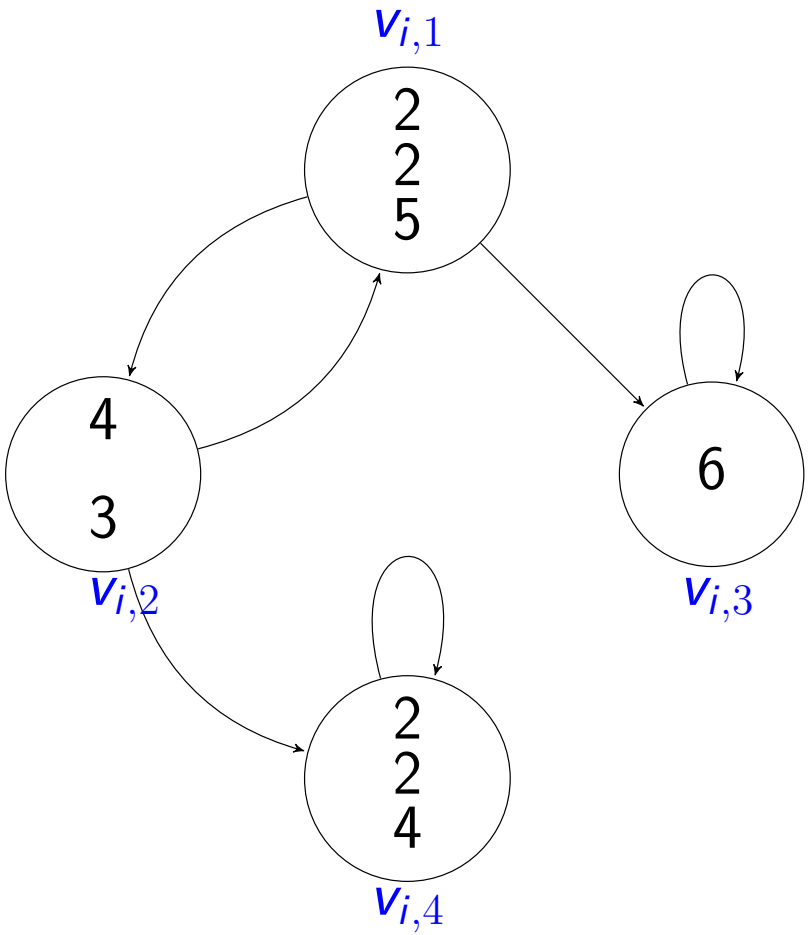
La tâche τ

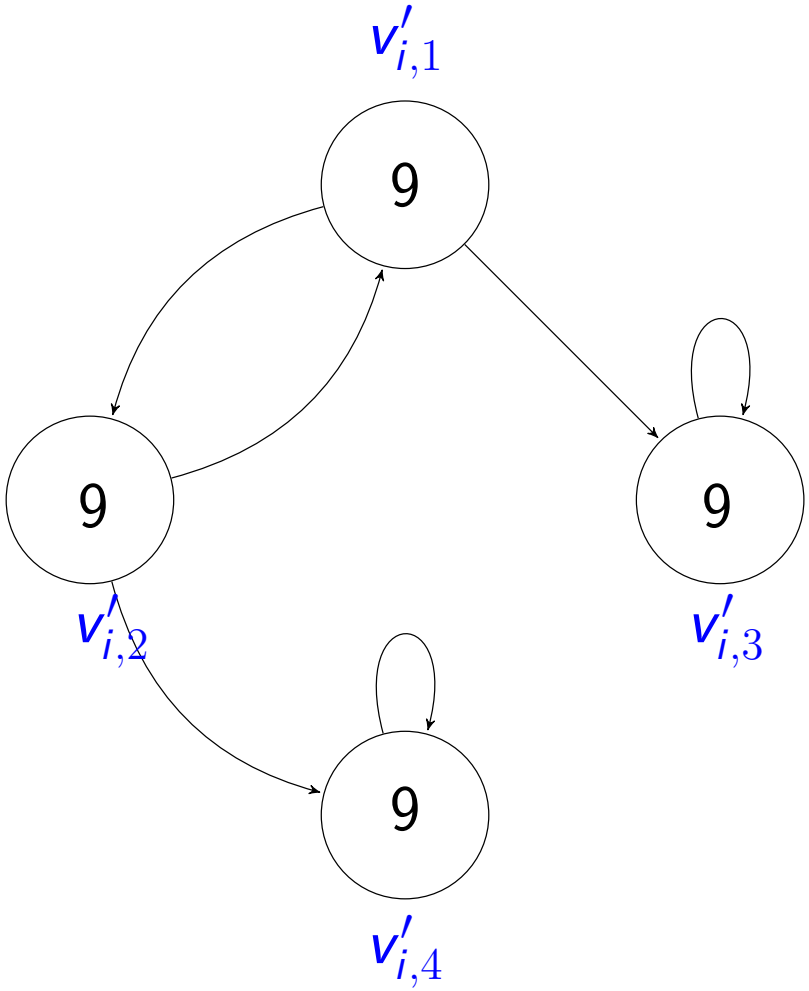
Tâche équivalente avec un thread



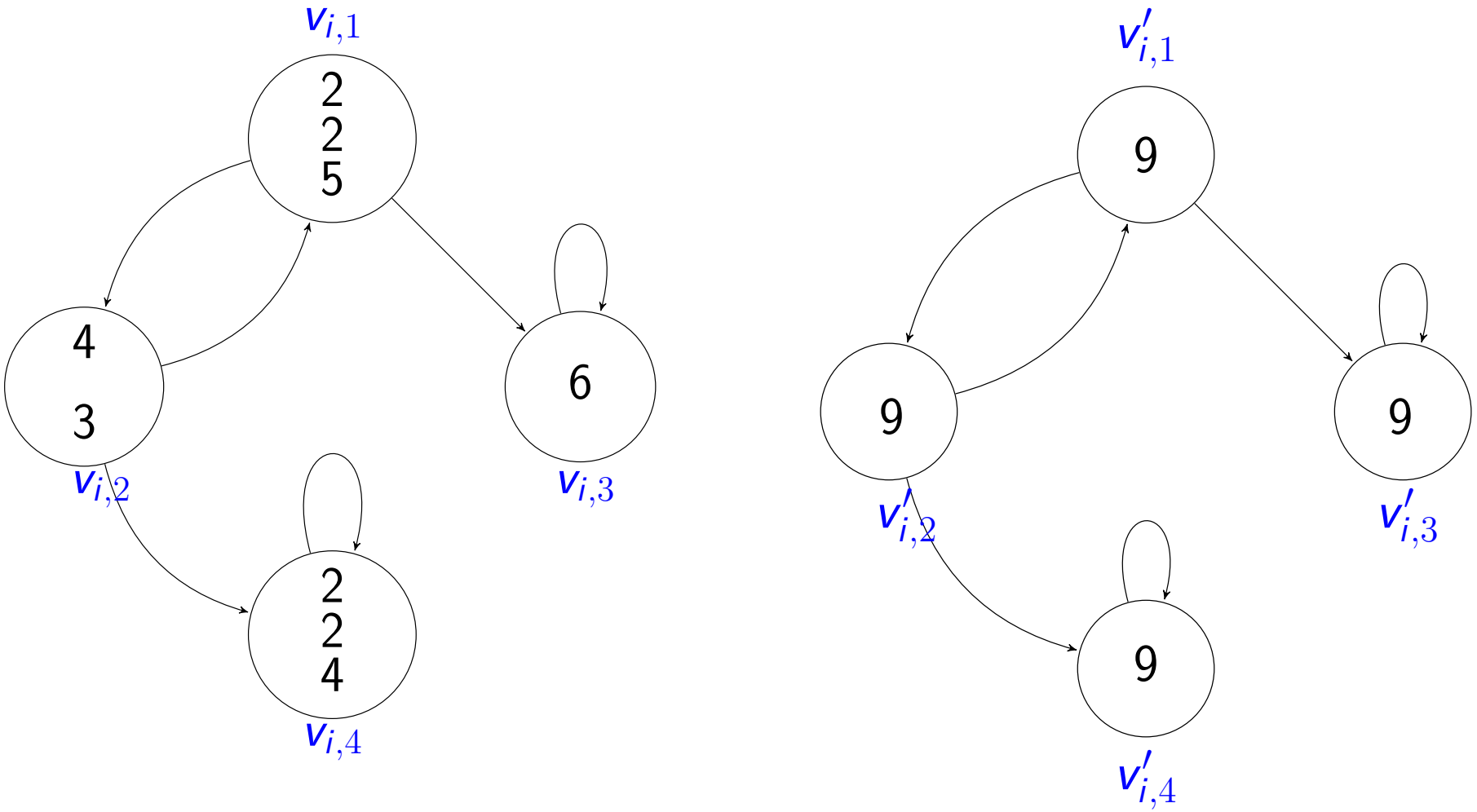


Tâche équivalente avec un thread



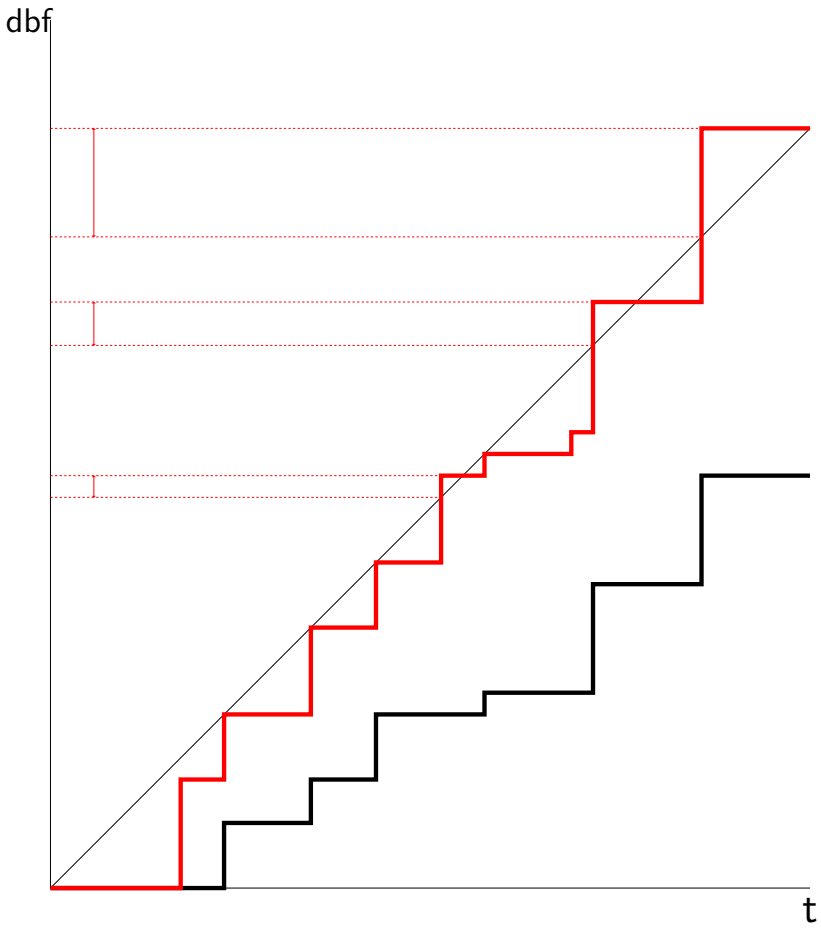


Tâche équivalente avec un thread

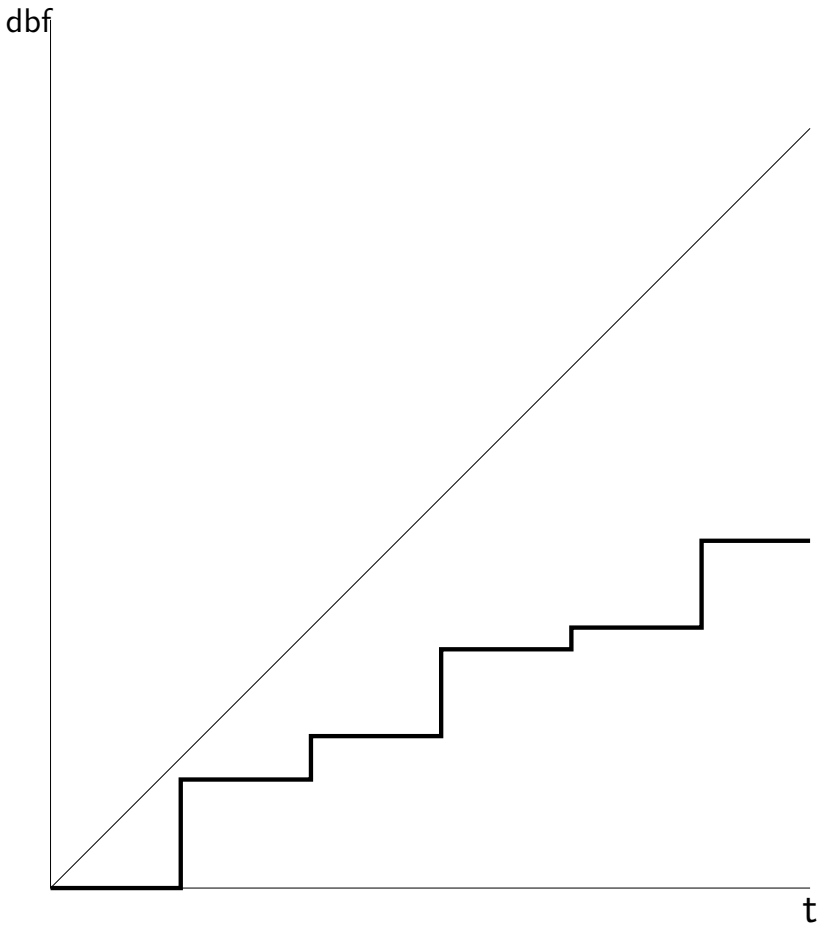


- Il s'agit de la conversion du modèle digraphe en modèle périodique de Liu et Layland.

Partitionnement

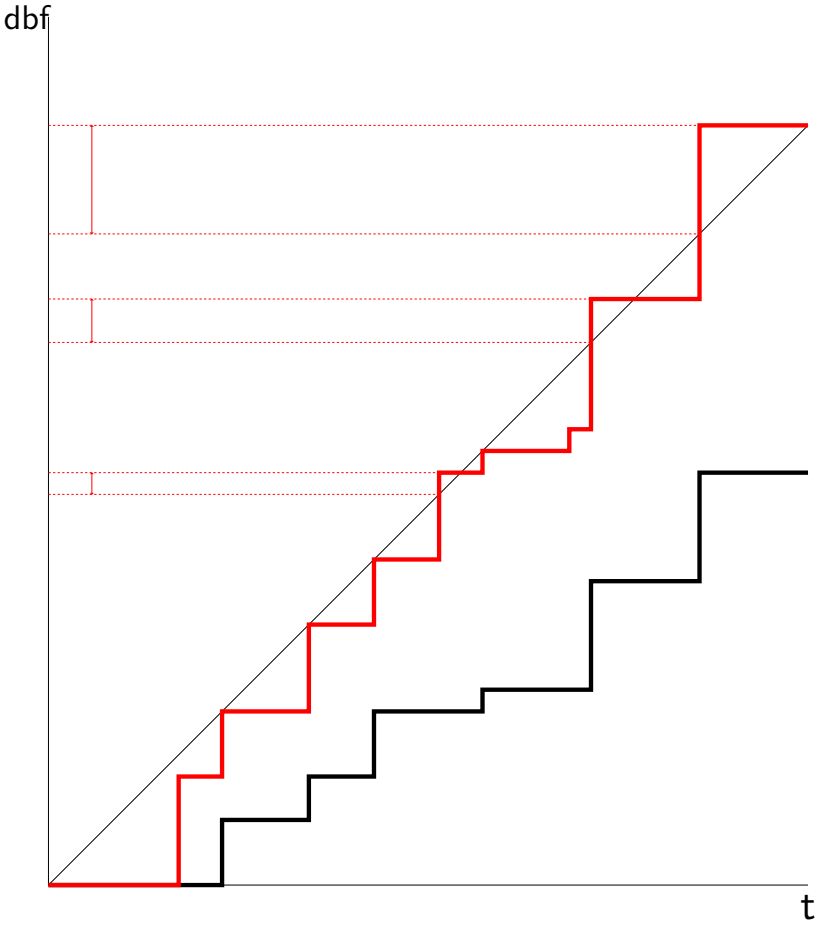


L'ensemble de tâches $\mathcal{T}$

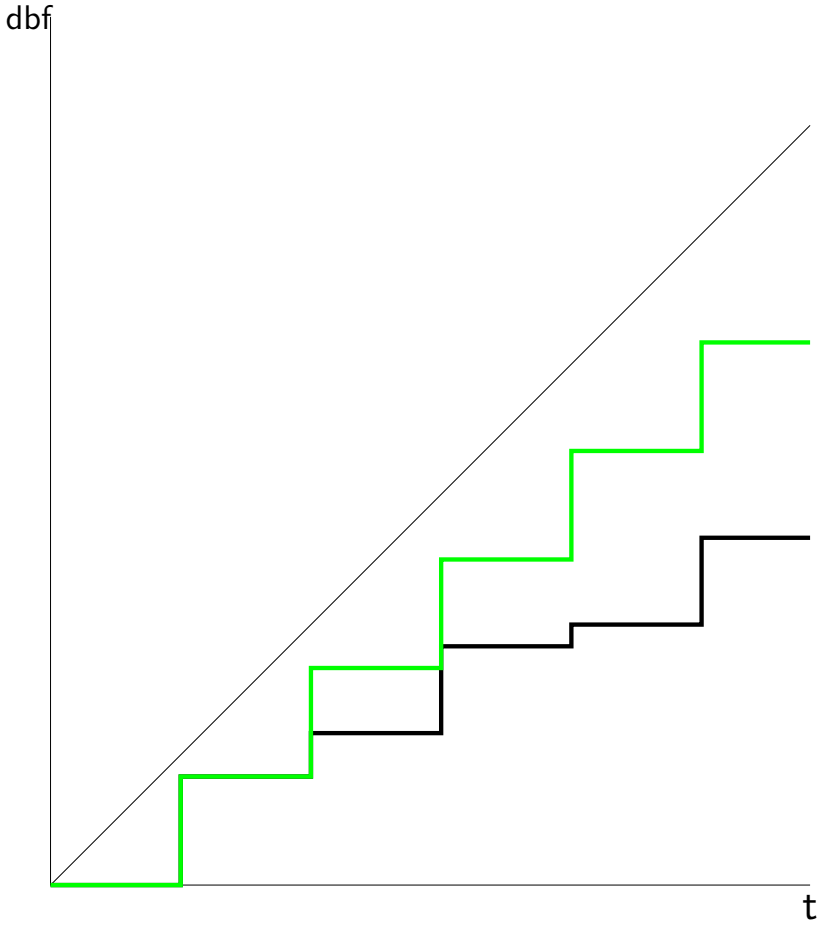


La tâche τ

Partitionnement

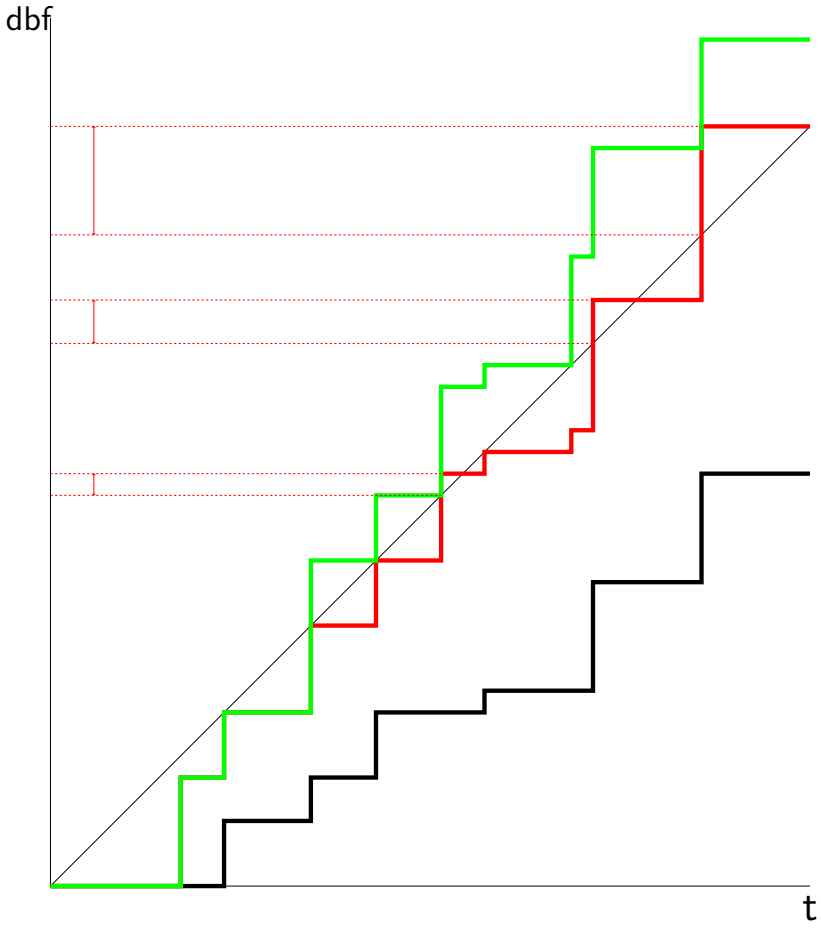


L'ensemble de tâches $\mathcal{T}$

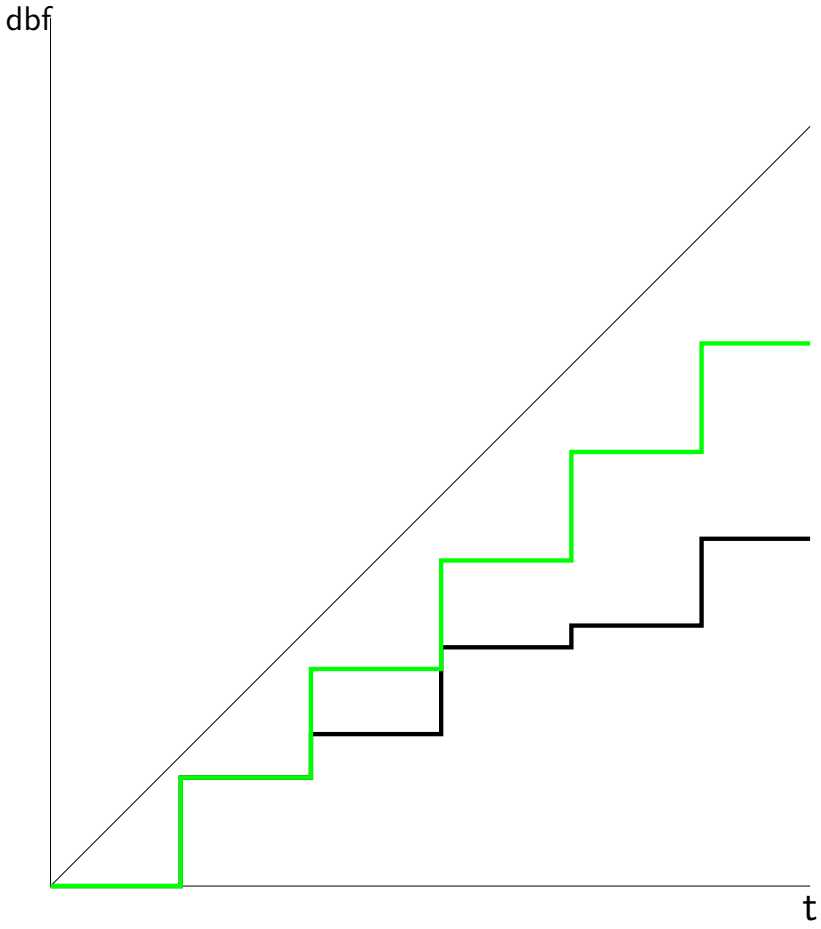


La tâche τ

Partitionnement

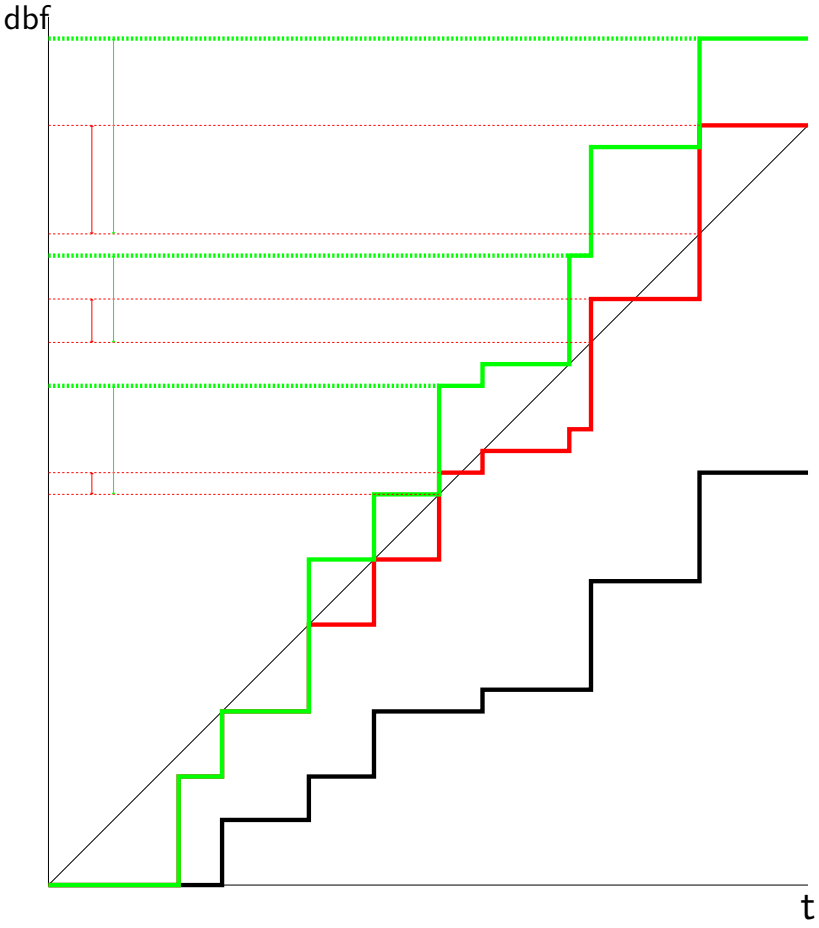


L'ensemble de tâches $\mathcal{T}$

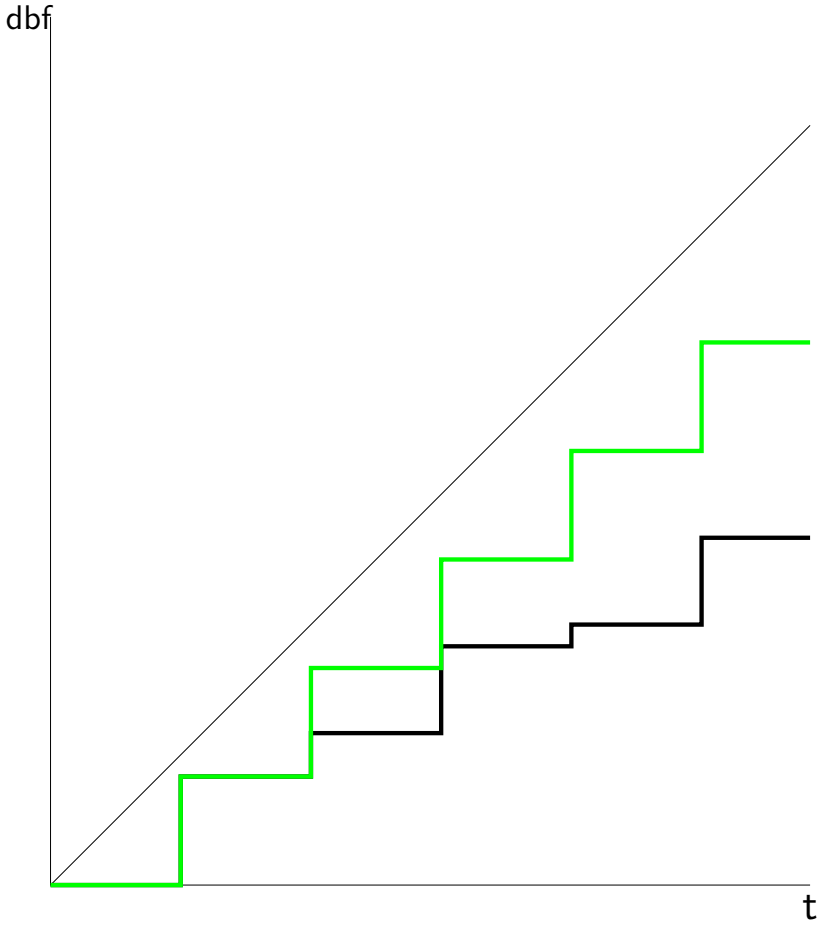


La tâche τ

Partitionnement



L'ensemble de tâches $\mathcal{T}$

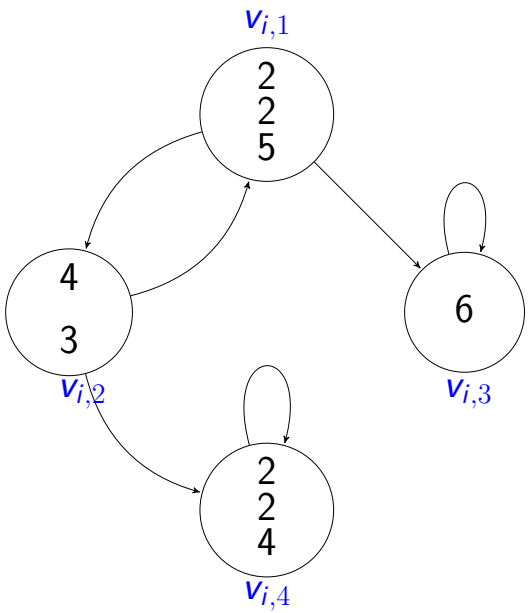


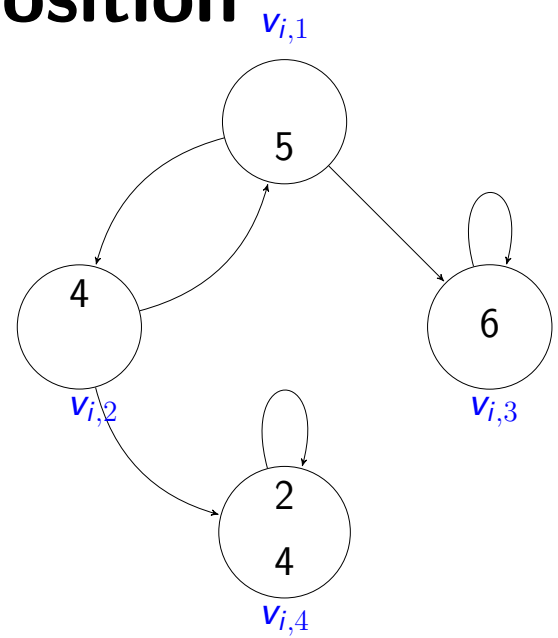
La tâche τ

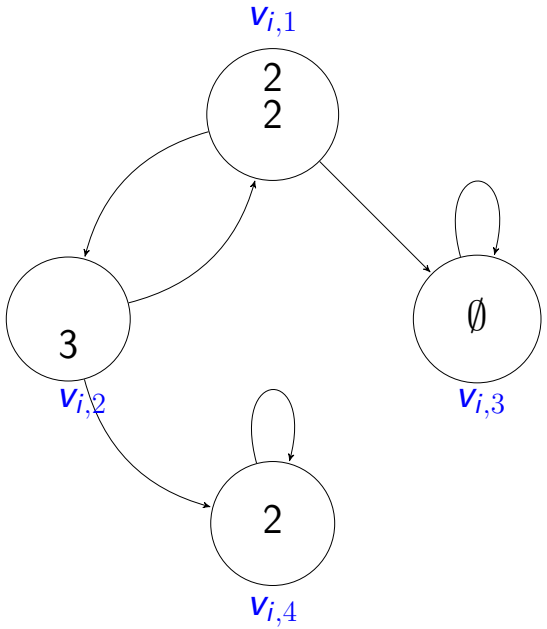
Un exemple de décomposition

Supposons que :

- *excess* $= 3$.
- la tâche à décomposer est :







Algorithme d'allocation

La sélection de fréquence

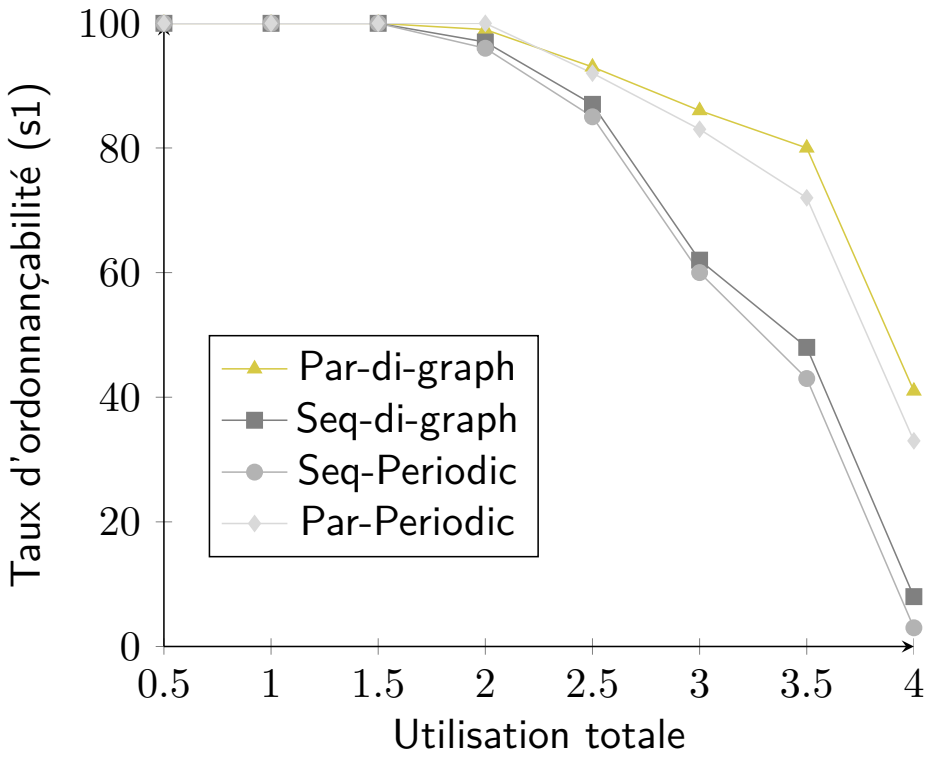
L'algorithme de sélection de fréquence est un algorithme glouton :

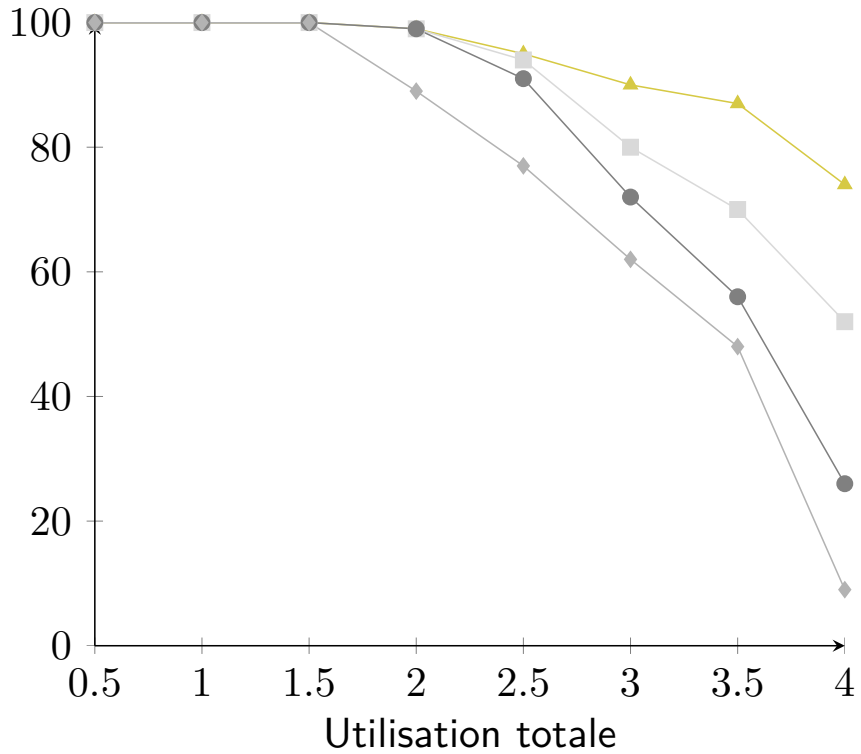
1. Sélectionner une fréquence
2. Vérifier l'ordonnançabilité
3. Si le test échoue, augmenter la fréquence, si non succès

Protocole expérimental

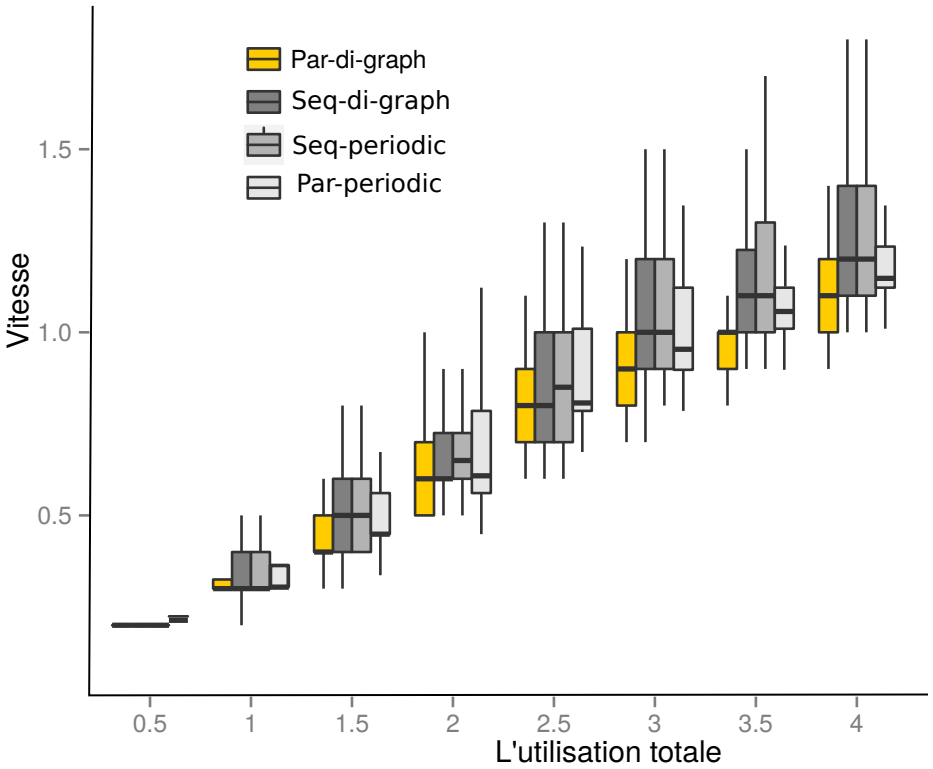
- La plateforme contient 4 coeurs
- Utilisation totale variée entre 0.5 et 4 par saut de 0.5
- Comparer contre :
  - Le modèle séquentiel de Liu et Layland (Seq-periodic)
  - Une parallélisation de Liu and Layland (Par-periodic)
  - Le modèle séquentiel de digraphe (Seq-di-graph)
- Scénarios :
  - Petite variabilité d'un état à un autre (± 0.1)
  - Grande variabilité d'un état à un autre (± 0.3)

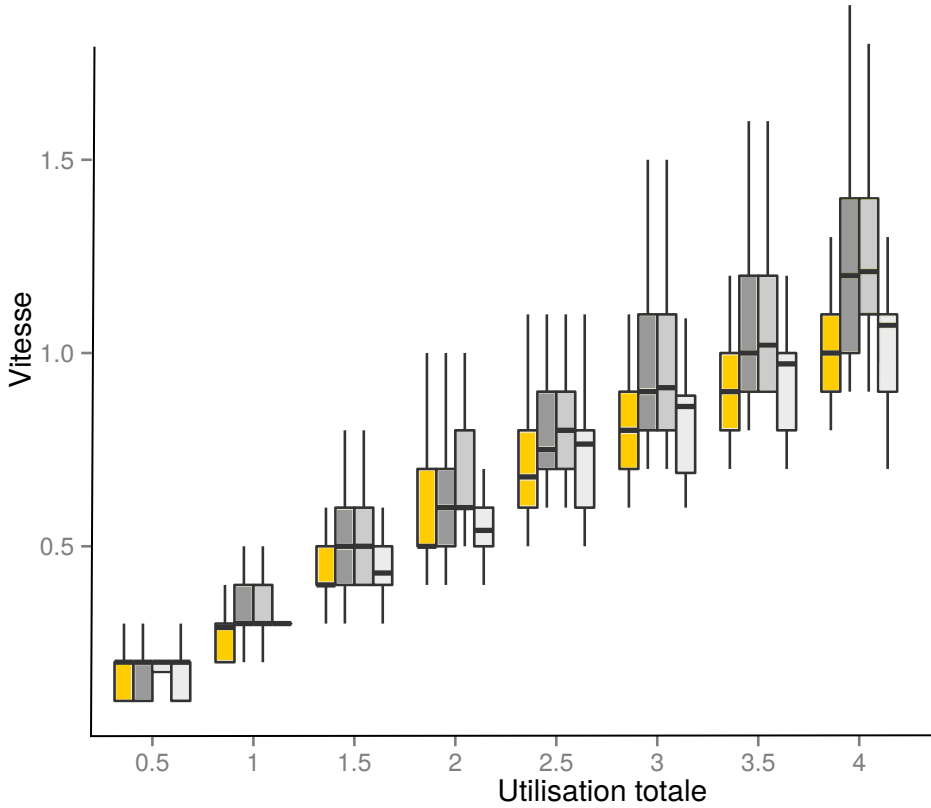
Le taux d'ordonnançabilité : S_1 vs S_2



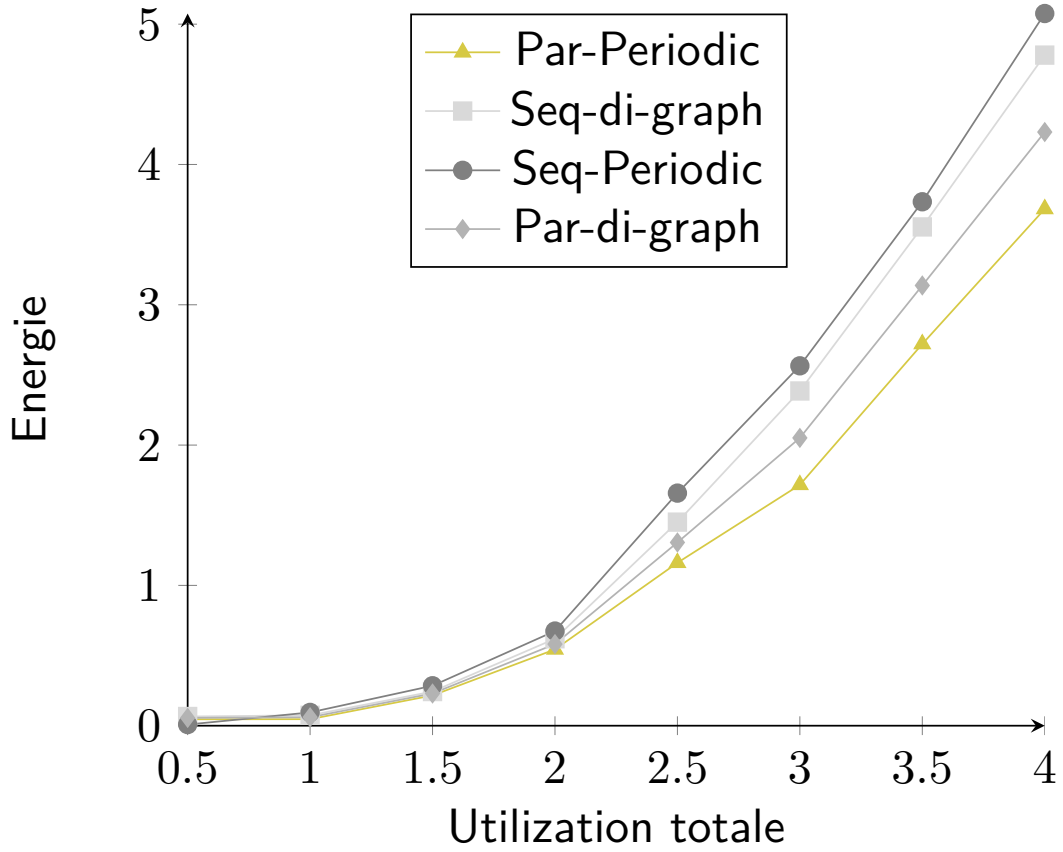


Minimisation de vitesse : S1 vs S2





Minimisation de consommation d'énergie : S2



Plan

Conclusion

J'ai proposé :

- Un modèle de temps et de consommation d'énergie basé sur des benchmarks sur une plateforme hétérogène.
- Deux modèles de tâches réalistes :
  - Le modèle CPM : modélise une tâche parallèle par un ensemble de cut-points.
  - Le modèle digraphe : modélise une tâche parallèle par un digraphe
- Deux tests d'ordonnançabilité ont été proposés pour analyser les modèles.

Perspectives

- Implanter les modèles proposés sur un API comme OpenMP
- Étendre le modèle digraphe avec des inter-arrivals et échéances arbitraires (travail en cours)
- Fusionner le modèle CPM et digraphe pour faire un modèle très expressif.
- Introduire le changement dynamique de la fréquence au problème.
- Utiliser les approximations de dbf pour réduire la complexité du modèle INLP

Travail scientifique

- H.E. Zahaf et al. "Modelling parallel task with Di-Graphs", RTNS'2016, Brest France, 19-21/10,2016.
- H.E. Zahaf et al. "Energy-Efficient Scheduling for Moldable Real-Time Tasks on Heterogeneous Computing Platforms", Under Revision Journal Of System Architecture
- H.E. Zahaf et al. "Modelling the Energy Consumption of Soft Real-Time Tasks on Heterogeneous Computing Architectures", EEHCO'2016, prague, January 15-16, 2016
- H.E. Zahaf, R. Olejnik, G. Lipari, A.E Benyamina , "Energy-aware moldable real-time task scheduling on uniform architectures", EDiS'2015, November 15-16, 2015
- H.E. Zahaf et al. "Intensive Real-Time Task Scheduling on Uniform Multiprocessors", (MOMA Journal), Vol 2, Issue 12014, Pages 3-13,
- H.E. Zahaf, A.E. Benyamina, R. Olejnik "Intensive Real-Time Task Scheduling on Uniform Multiprocessors",The 2 nd international Workshop on Mathamatics and Computer Science IWMCS'2014, december 1-3, 2014, Tiaret, Algeria.