Algorithmes à front d'onde et accès transparent aux données

Pierre-Nicolas Clauss

Équipe-projet AlGorille

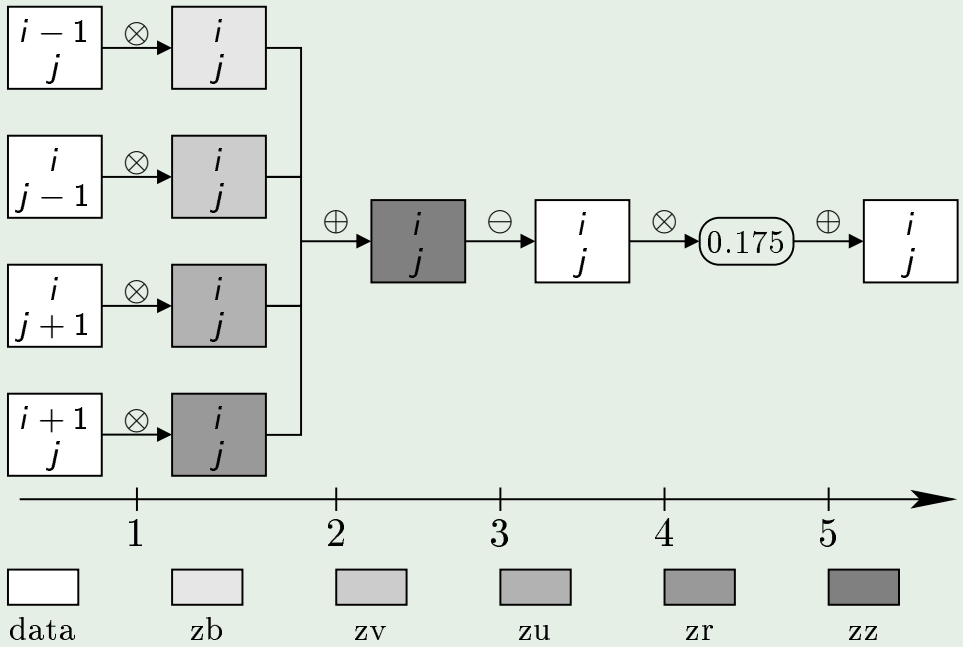
Plan général

Plan

Benchmark *LINPACK*

Livermore Kernel 23

pour $l = 1$ à $Loops$ **faire**
  pour $j = 2$ à $M - 1$ **faire**
    pour $i = 2$ à $N - 1$ **faire**
      $q \leftarrow$ data$[j][i-1] *$ zb$[j][i] +$ data$[j-1][i] *$ zv$[j][i]$
        $+$data$[j+1][i] *$ zu$[j][i] +$ data$[j][i+1] *$ zr$[j][i]$
        $+$zz$[j][i]$
      data$[j][i] \leftarrow$ data$[j][i] + 0.175 * (q -$ data$[j][i])$
    fin pour
  fin pour
fin pour

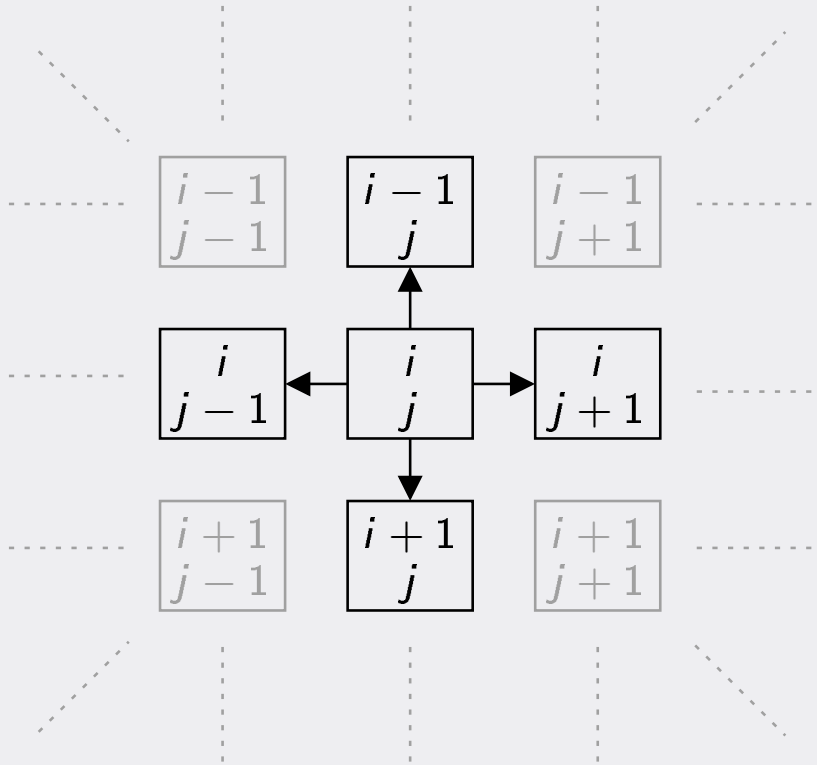


Plan

Dépendance de données

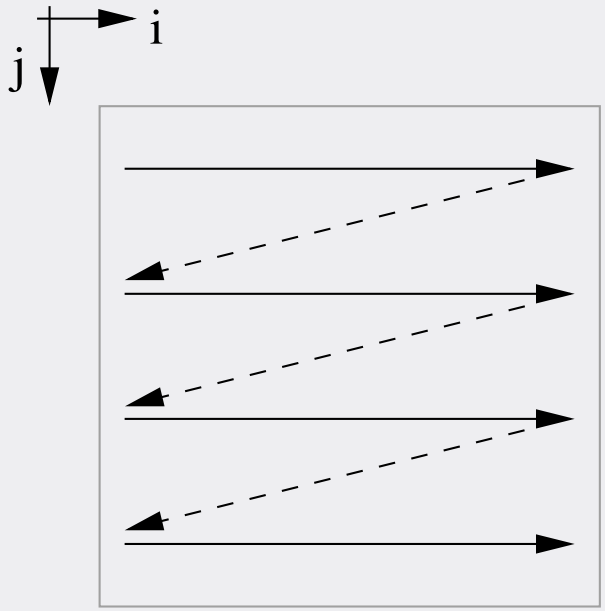
Caractéristiques des algorithmes à front d'onde

- Espace de données discret
- Calculs locaux
- Calculs itératifs
- Calculs potentiellement Out-of-Core



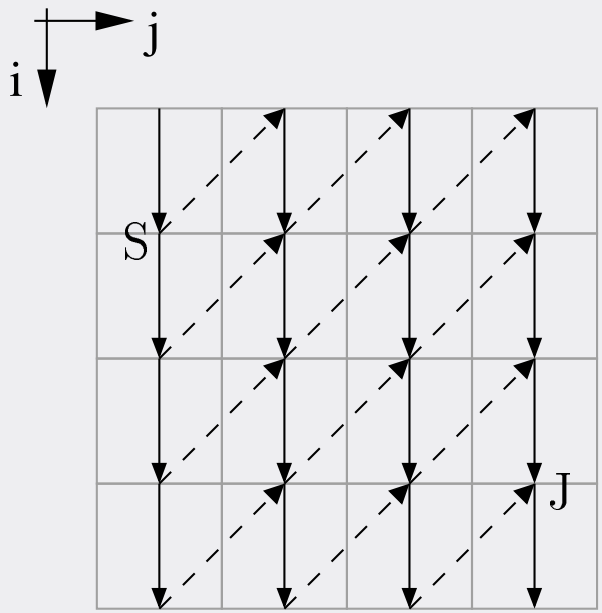
Modèles d'exécution

Exécution séquentielle



- Parcours lignes par lignes

Exécution parallèle

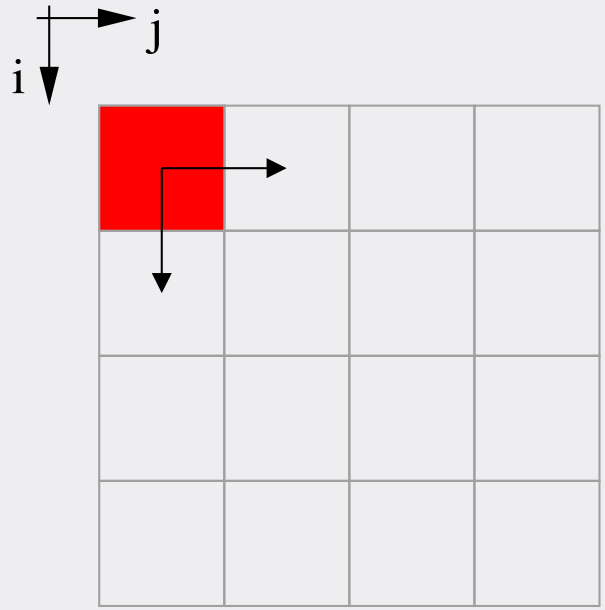


- Parcours blocs par blocs

Onde de calcul

Progression des calculs

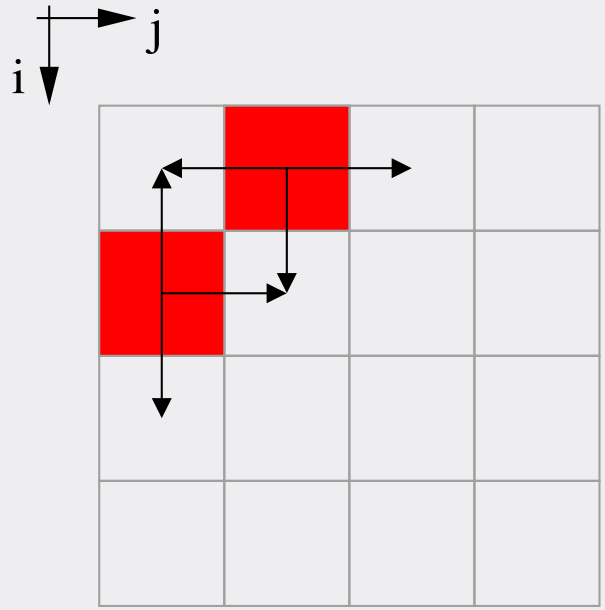
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

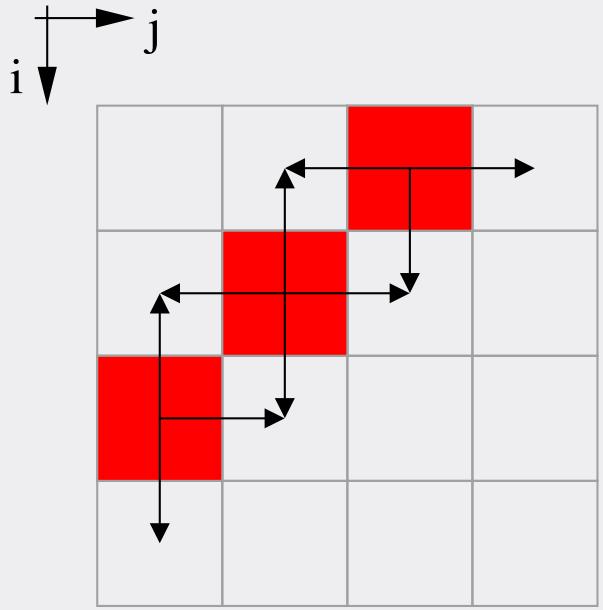
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

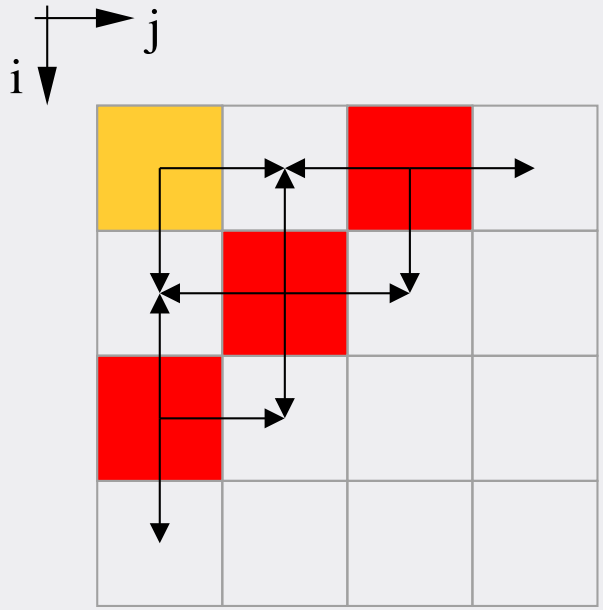
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

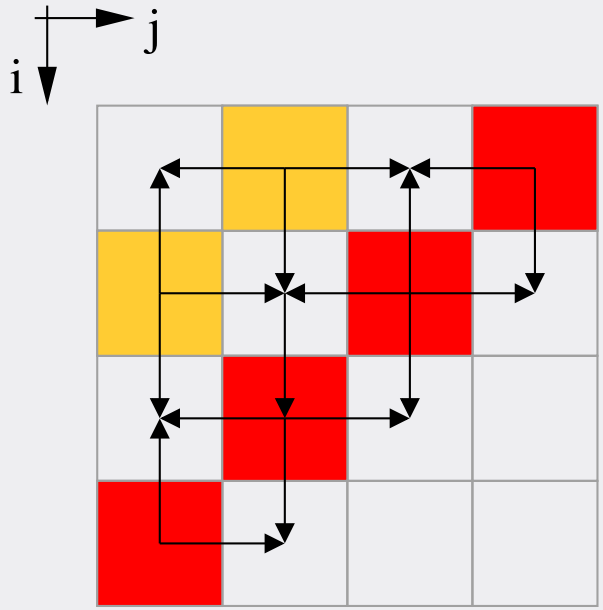
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

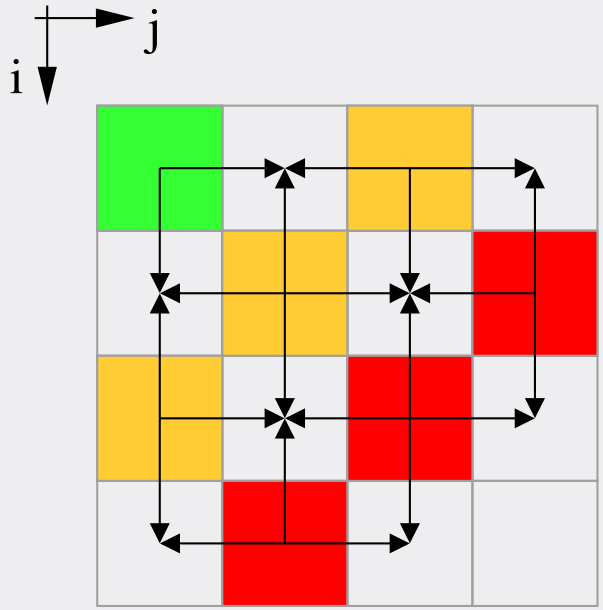
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

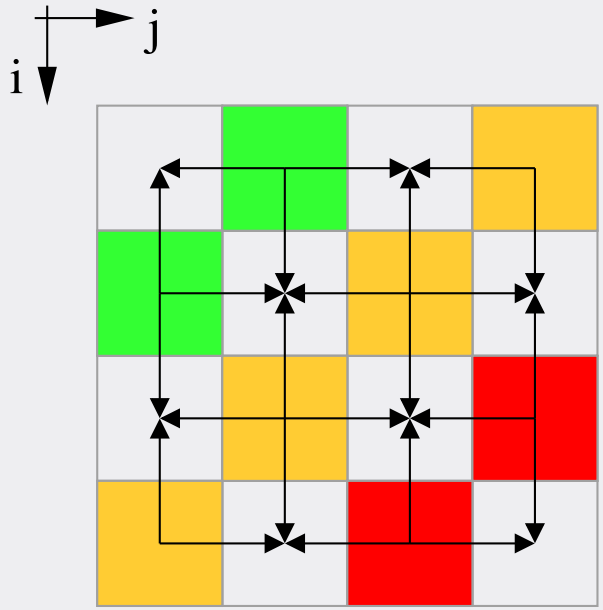
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

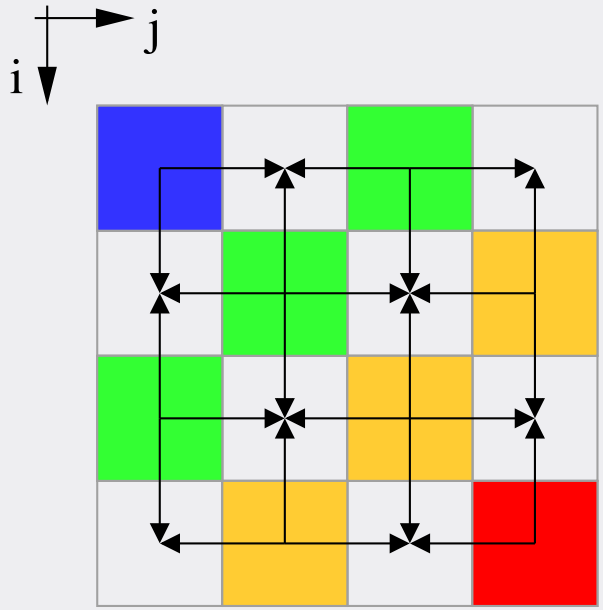
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Onde de calcul

Progression des calculs

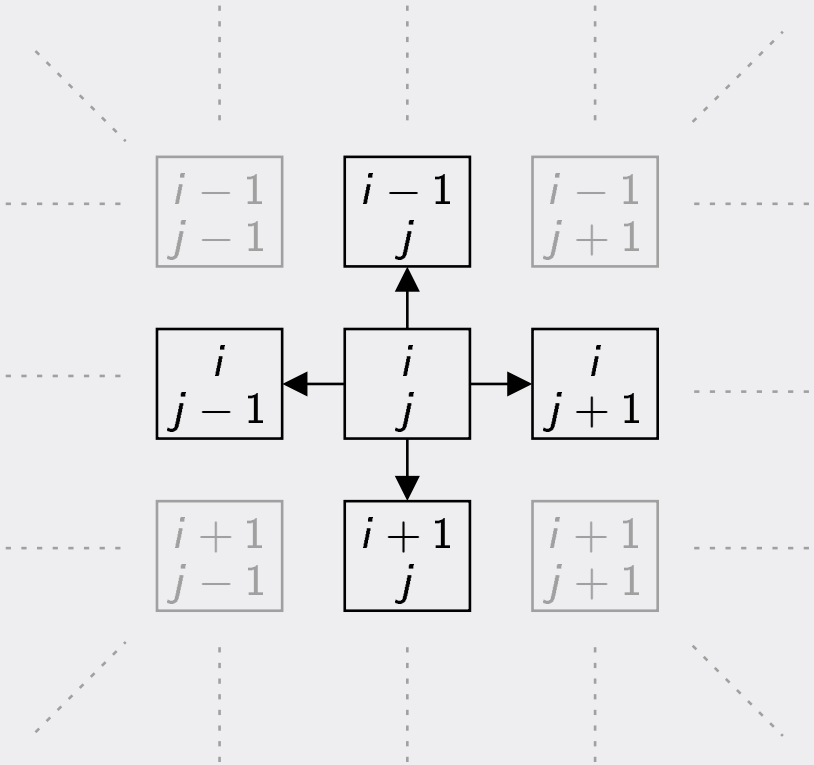
- Phase transitoire
- Enchaînement des vagues
- Régime permanent



Plan

Accès aux éléments

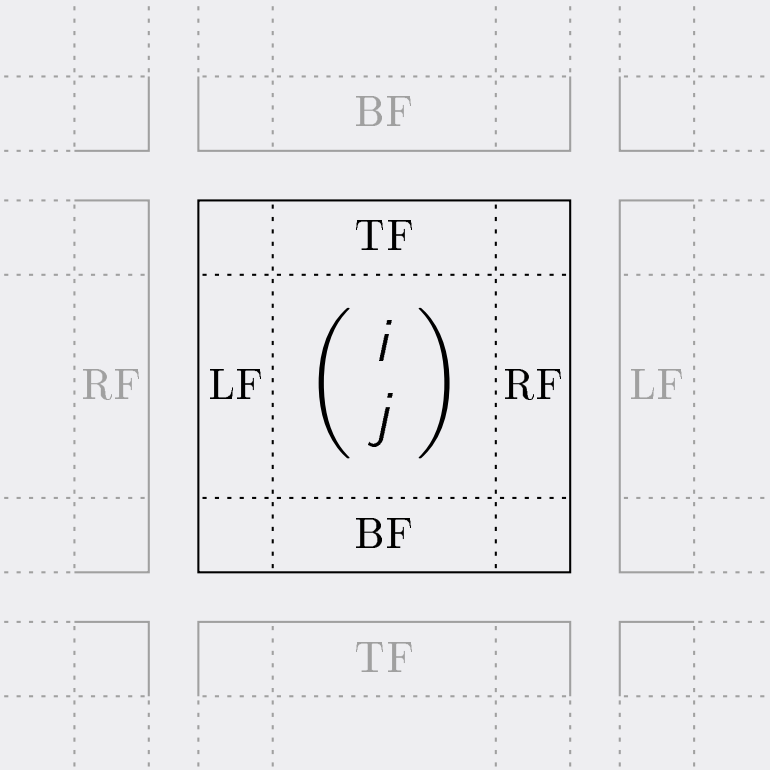
Dépendance de données



- Lecture des 4 voisins

Accès aux blocs

Communications

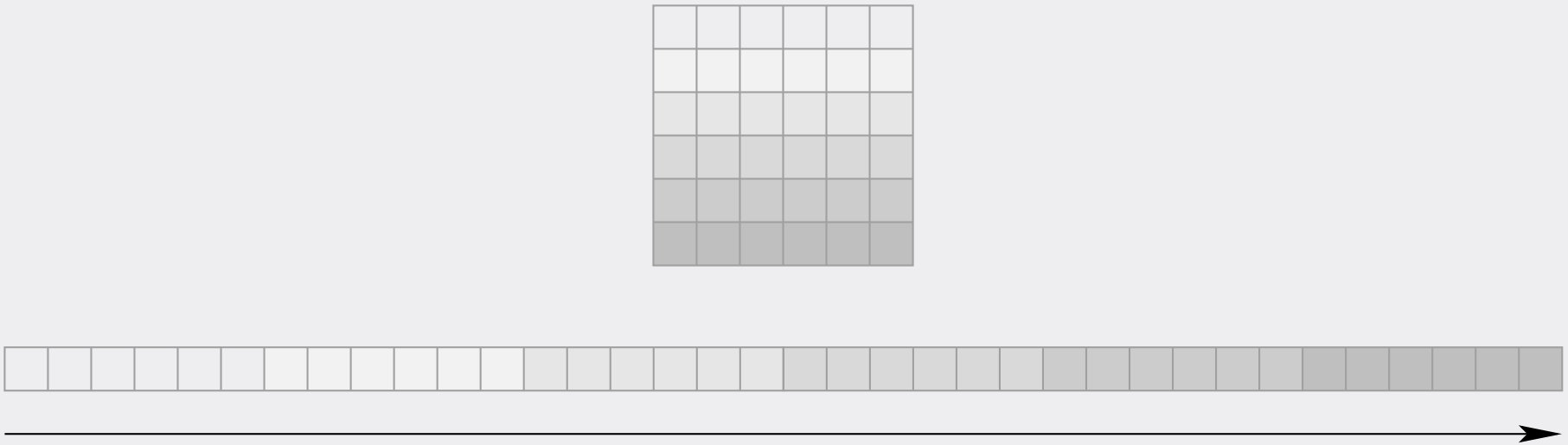


- Regroupement en blocs
- 4 frontières par bloc
- Lecture des frontières voisines

Plan

Vers un stockage optimisé

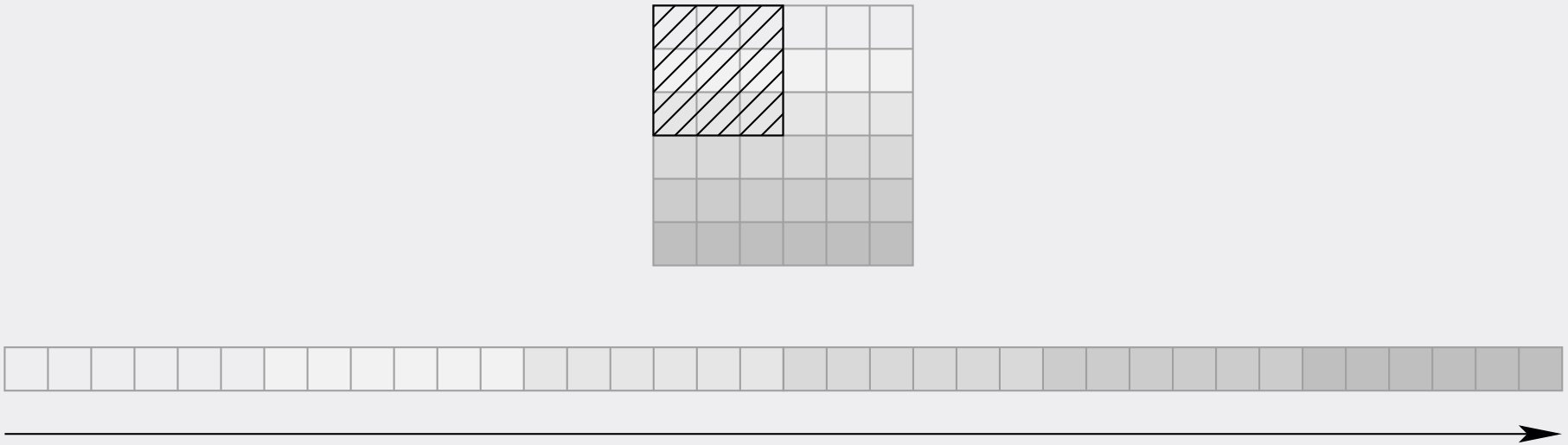
Stockage par lignes «canonique»



- Stockage d'origine
- Problème : Accès non-contigus aux blocs

Vers un stockage optimisé

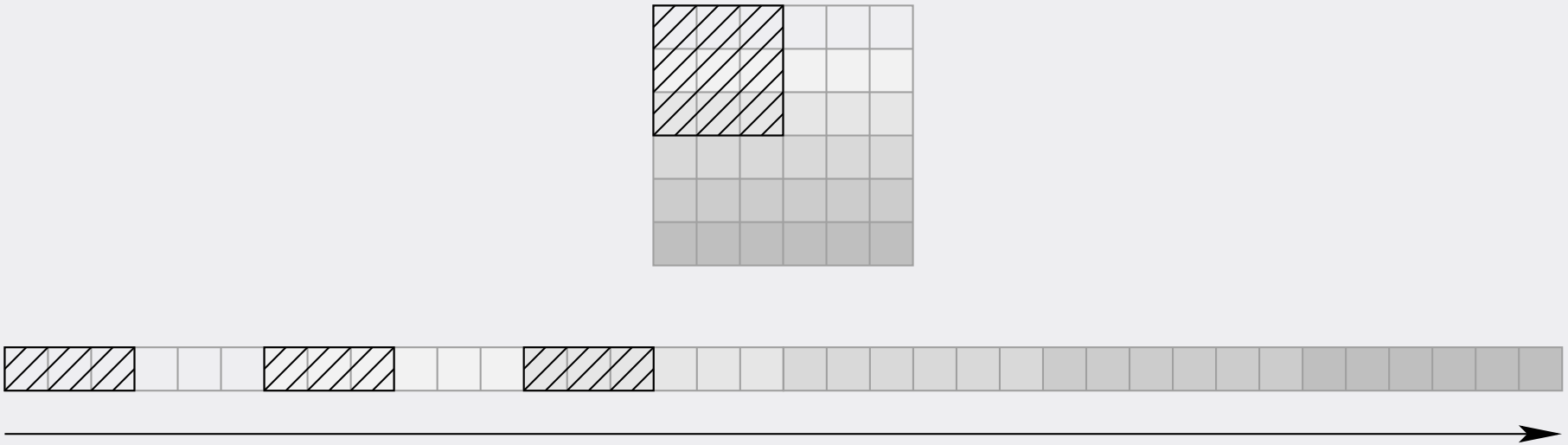
Stockage par lignes «canonique»



- Stockage d'origine
- Problème : Accès non-contigus aux blocs

Vers un stockage optimisé

Stockage par lignes «canonique»



- Stockage d'origine
- Problème : Accès non-contigus aux blocs

Approches possibles

Approche applicative

- Philip Rhodes *et al.*, 2005
- Améliorer la politique de préchargement
- Utilisation de la connaissance du sens de parcours
- Effort contradictoire
  - Avec la politique du système d'exploitation
  - Avec une parallélisation souple et performante

Approche système

- Eddy Caron *et al.*, 1999
- Modification du gestionnaire de mémoire virtuelle
- Bonnes performances
- Fortement lié à une application

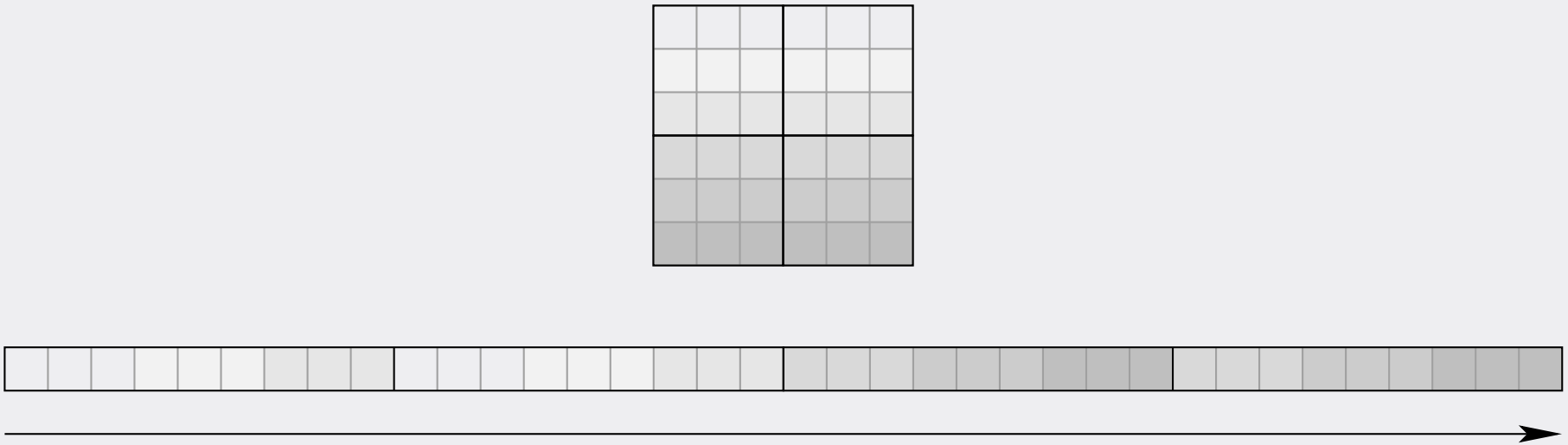
Approche *Out-of-core*

Gestion applicative du problème

- Eddy Caron, Frédéric Suter *et al.*, 2005
- Superposition des calculs et communications sur les E/S
- Pas de gestion de l'agencement des données sur disque

Vers un stockage optimisé

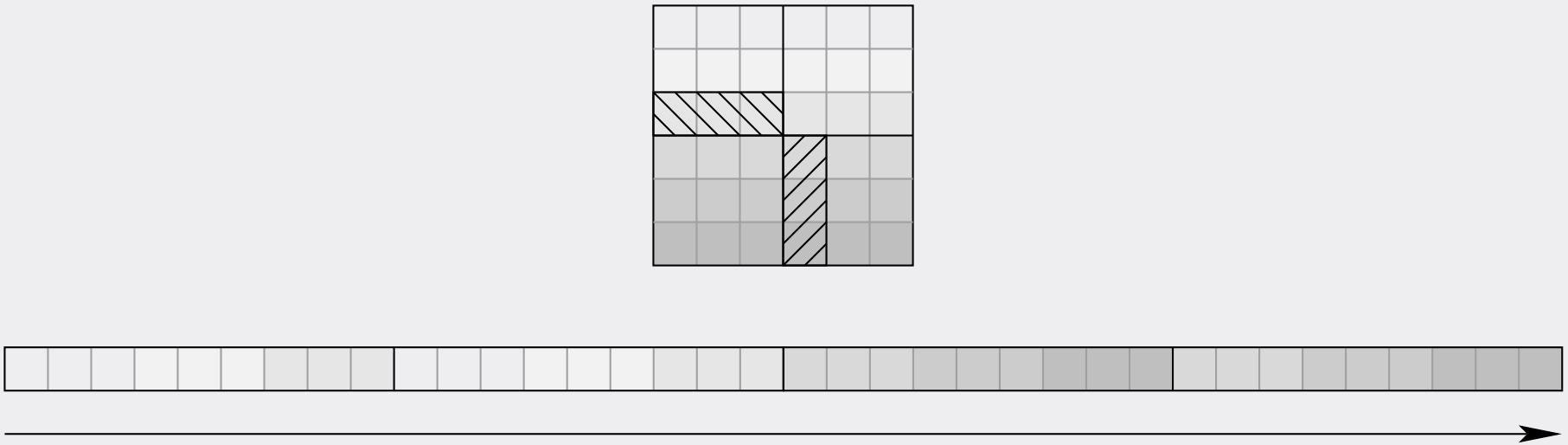
Stockage par blocs



- Siddartha Chatterjee *et al.*, 1999
- Accès contigus aux blocs
- Stockage interne des blocs en lignes
- Problème : Accès non-contigus aux frontières verticales

Vers un stockage optimisé

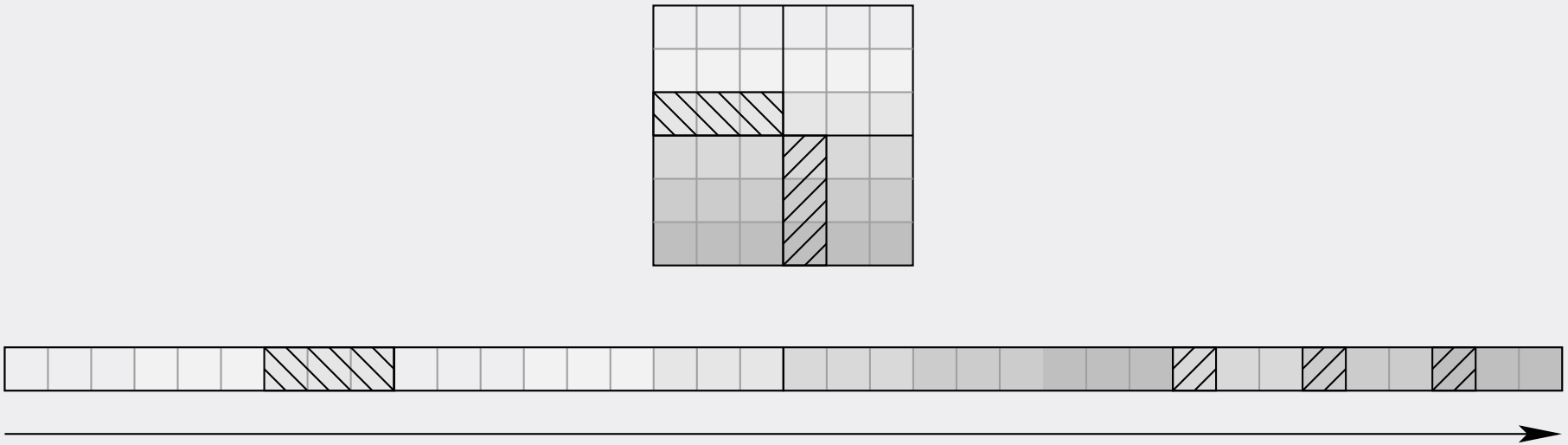
Stockage par blocs



- Siddartha Chatterjee *et al.*, 1999
- Accès contigus aux blocs
- Stockage interne des blocs en lignes
- Problème : Accès non-contigus aux frontières verticales

Vers un stockage optimisé

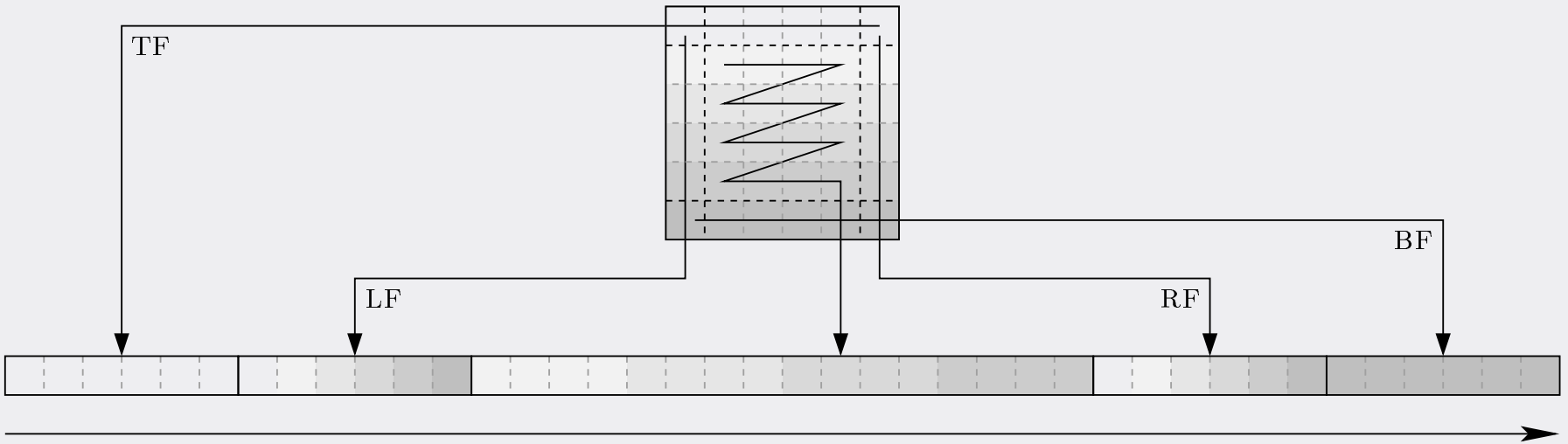
Stockage par blocs



- Siddartha Chatterjee *et al.*, 1999
- Accès contigus aux blocs
- Stockage interne des blocs en lignes
- Problème : Accès non-contigus aux frontières verticales

Vers un stockage optimisé

Stockage étendu : détail d'un bloc



- Accès contigus aux blocs
- Accès contigus aux frontières
- Duplication des éléments communs aux frontières

Inconvénients du stockage étendu

Propagation des copies

- Nécessité de propager une copie de chaque modification d'un coin d'un bloc
- Les langages modernes permettent une implémentation qui cache cette nécessité

Augmentation du volume de données

- Taille des données dans le stockage étendu :

$$T' = T * \left(1 + \frac{4 * F_h * F_v}{R * C}\right) \text{ avec } F_h \leq R \text{ et } F_v \leq C$$

- Facteur de surcoût de l'ordre du pourcent dans un cas réel

Changement d'agencement

- Pré- et post-traitements

Plan

Problématique

Caractéristiques

- Algorithme à front d'onde : accéder aux données voisines
- Accès exclusifs ou inclusifs
- Nécessité de synchroniser entre les blocs de données

Modèle générique

- Calculs sur des données prédécoupées en *lieux*
- Synchronisation conforme à la sémantique des *Read-Write Locks* (RWL)
- Dépendances de données entre les tâches
- Possibilité de calculs itératifs

Ordered Read-Write Locks

Outils existants

- Read-Write Locks (Posix, 2004) (C. Wagner et F. Mueller, 2000)
- Capture atomique (Yehuda Afek *et al.*, 1993)
  - Centrée sur les opérations sans attentes
- Mutex avec FIFO (Robert Danek *et al.*, 2008)
  - Pas de différentiation entre les types d'accès

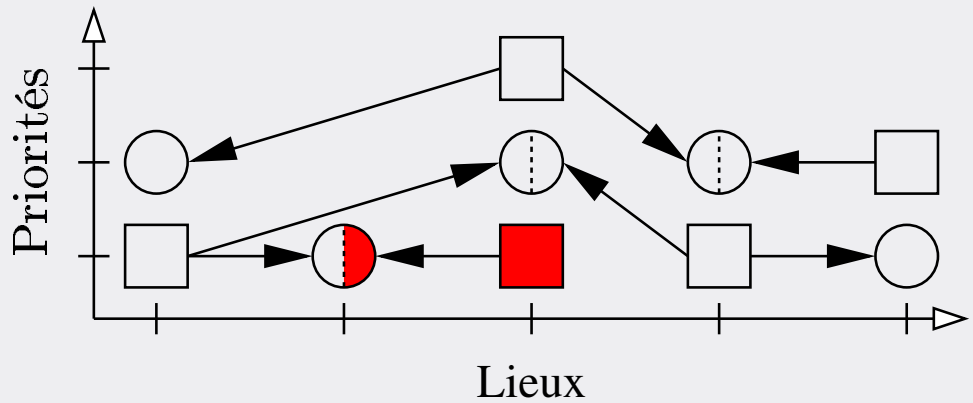
ORWL : Outil bas-niveau

- Accès concurrents en lecture (requêtes ◯)
- Accès exclusifs en écriture (requêtes □)
- Politique d'attente FIFO
- Orientés ressource
  - Accès à travers un *Lock Handle* (LH)
  - Accès pro-actifs controllés

Modélisation d'un système

Overlay

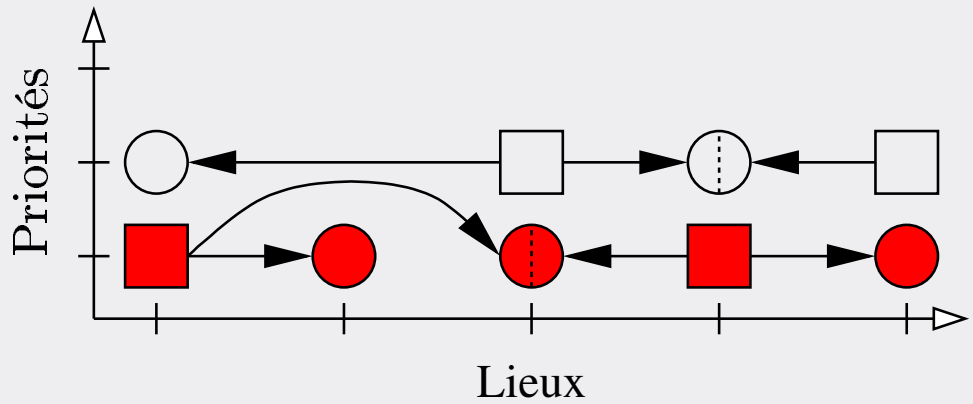
- Un ensemble de *lieux* verrouillables
- Un ensemble de tâches représentées par des requêtes connectées
- Exécution d'une tâche :
  1. Acquisition des requêtes
  2. Calculer
  3. Retirer les requêtes



Modélisation d'un système

Overlay

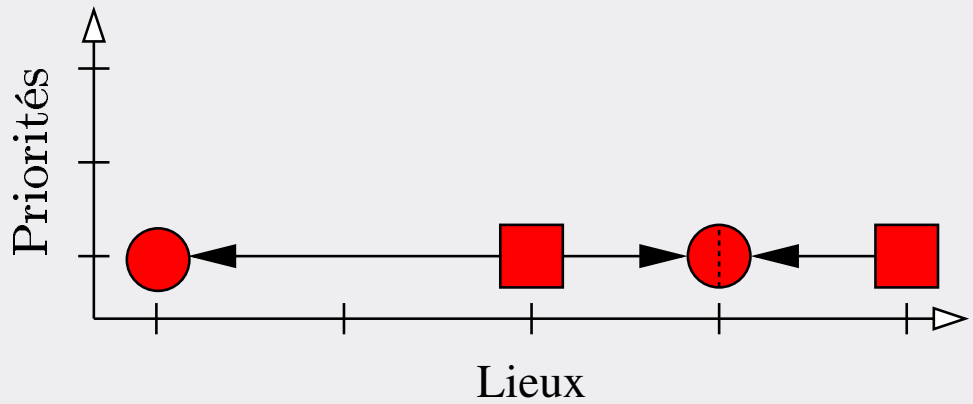
- Un ensemble de *lieux* verrouillables
- Un ensemble de tâches représentées par des requêtes connectées
- Exécution d'une tâche :
  1. Acquisition des requêtes
  2. Calculer
  3. Retirer les requêtes



Modélisation d'un système

Overlay

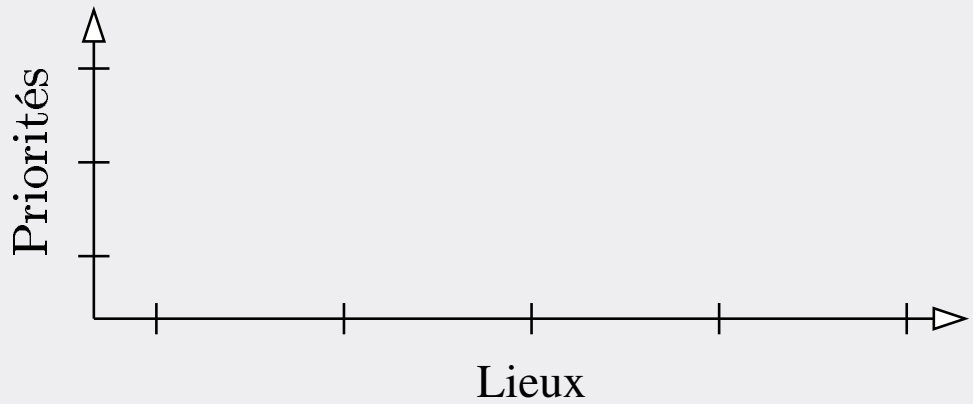
- Un ensemble de *lieux* verrouillables
- Un ensemble de tâches représentées par des requêtes connectées
- Exécution d'une tâche :
  1. Acquisition des requêtes
  2. Calculer
  3. Retirer les requêtes



Modélisation d'un système

Overlay

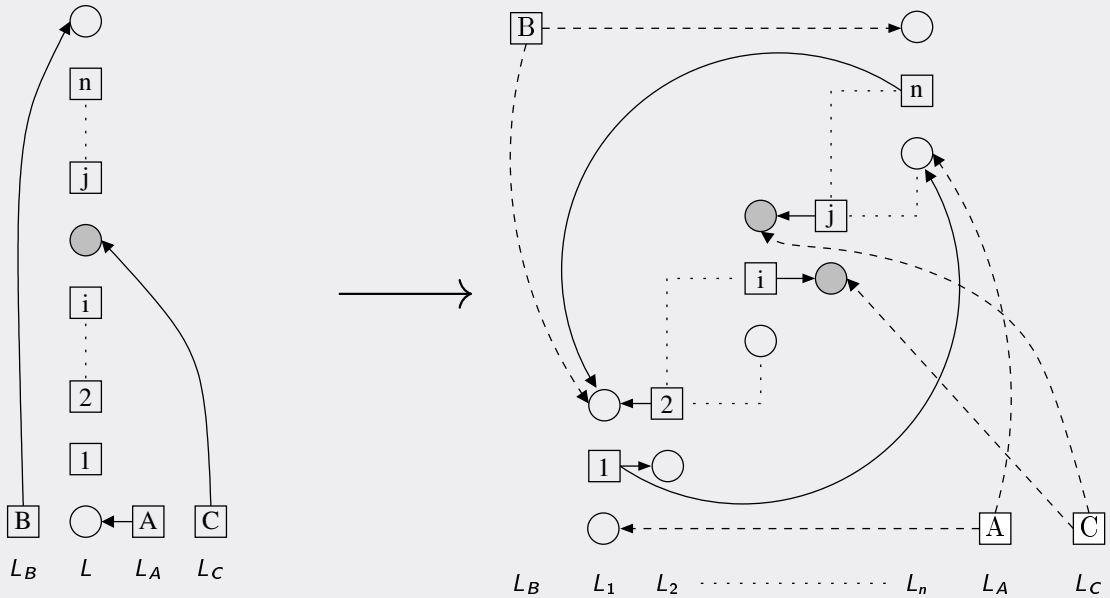
- Un ensemble de *lieux* verrouillables
- Un ensemble de tâches représentées par des requêtes connectées
- Exécution d'une tâche :
  1. Acquisition des requêtes
  2. Calculer
  3. Retirer les requêtes



Plan

Construction de la forme canonique

Identification des requêtes, des tâches et des lieux

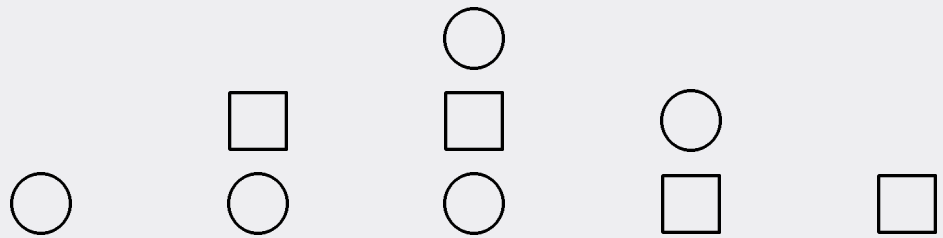


- Conserve l'ordre dans lequel les requêtes originales sont acquises
- Taille de l'overlay canonique linéaire par rapport à la taille de l'overlay d'origine

Avantage de la forme canonique

Combinaisons canoniques

- Implémentation simplifiée



- Preuves des propriétés du modèle

Plan

Structure du calcul itératif

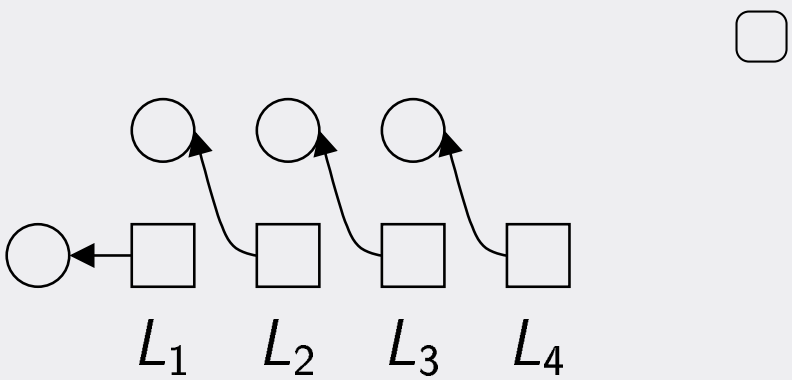
Tâches itératives

- Les tâches peuvent s'exécuter plusieurs fois
- L'exécution est controllée par un évènement externe
  - Stabilisation des calculs
  - Nombre d'itérations prédéfini
- Les ORWL permettent de poster de nouvelles requêtes similaires à celles déjà acquises

Modèle d'exécution des tâches itératives

Exemple d'évolution

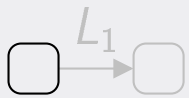
1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

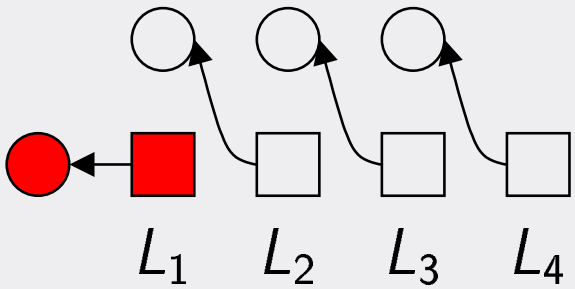


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

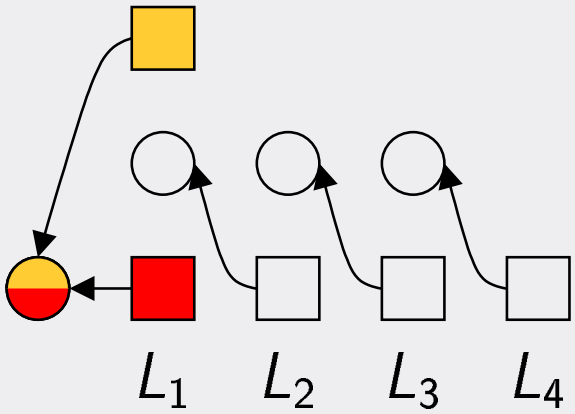


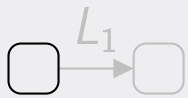


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

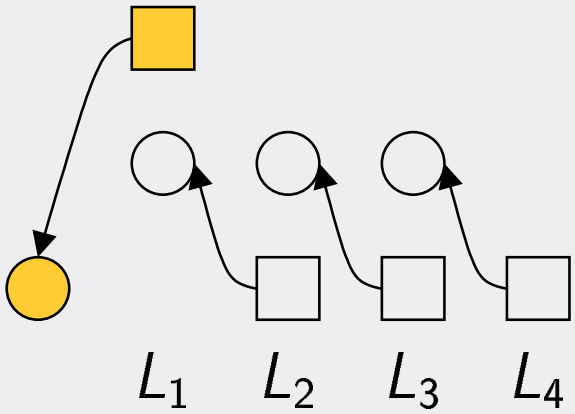


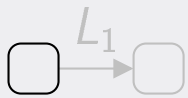


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

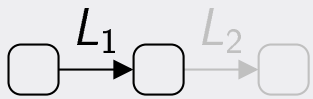


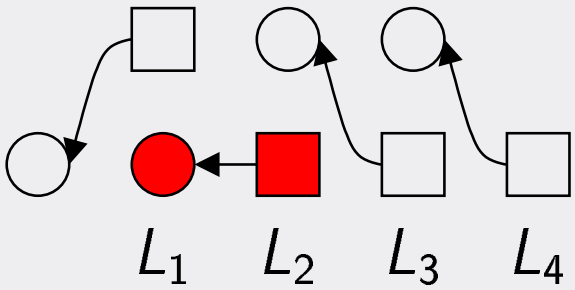


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

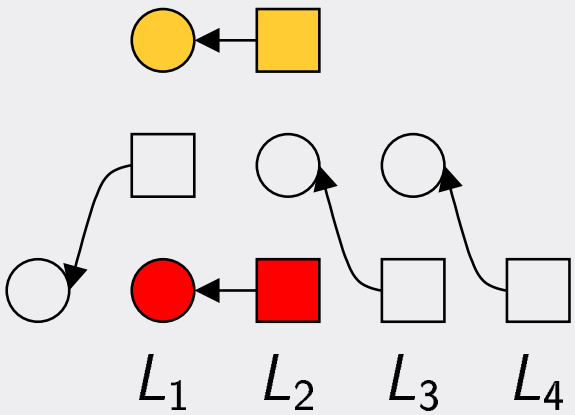


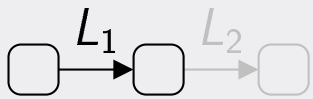


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

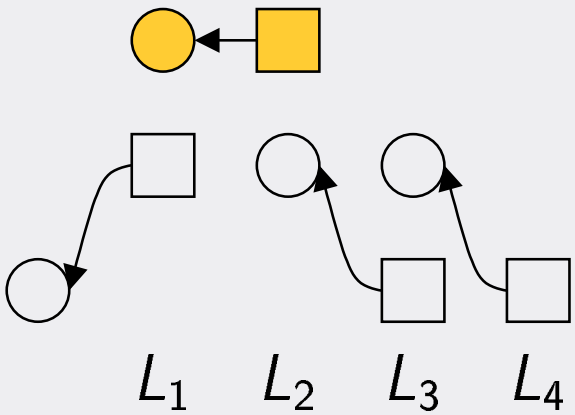


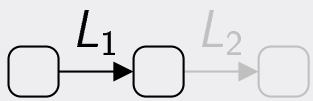


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

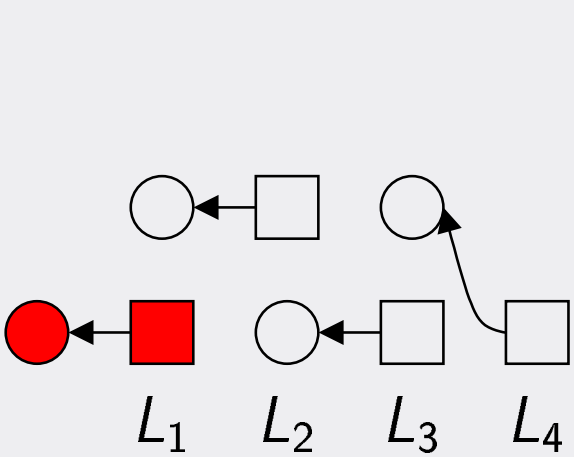


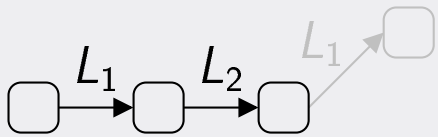


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

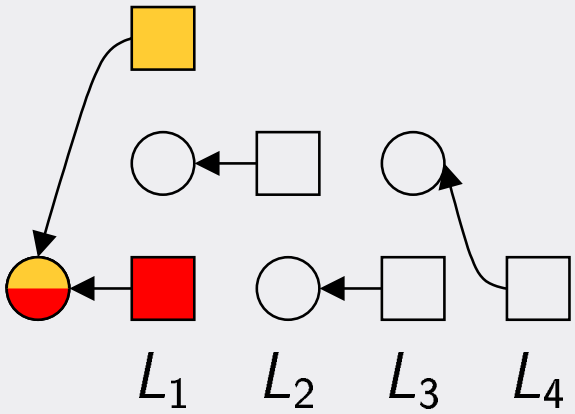


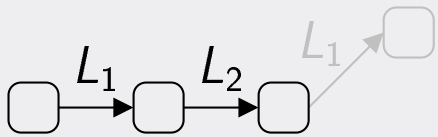


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

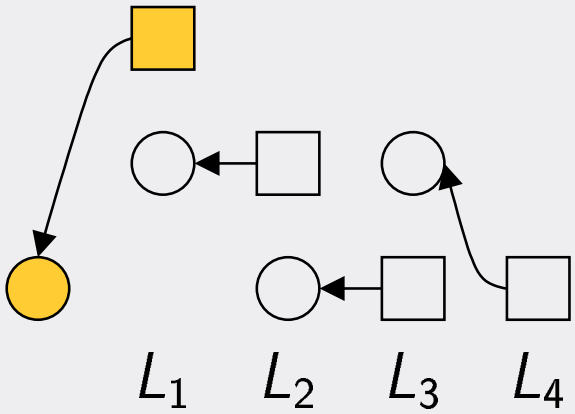


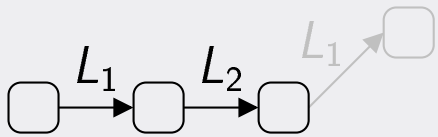


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

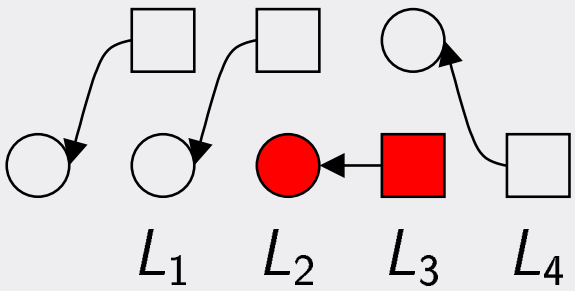


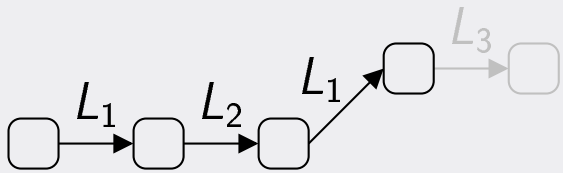


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

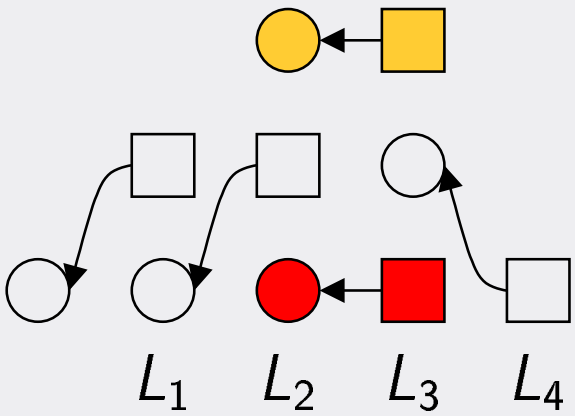


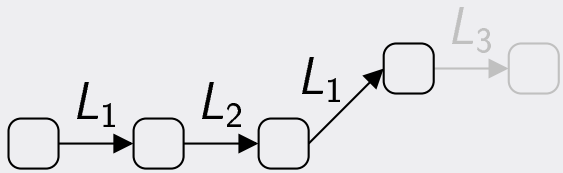


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

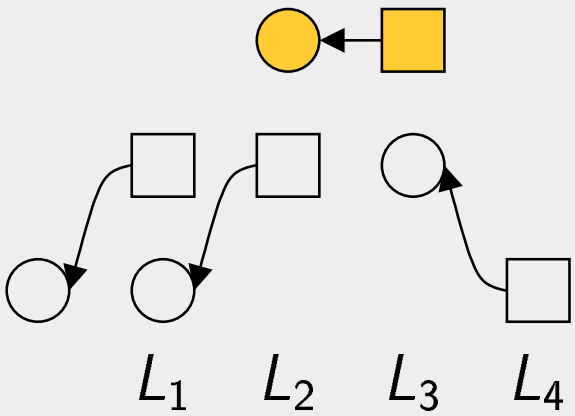


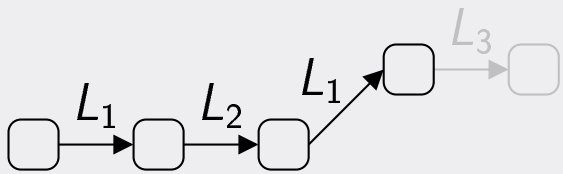


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

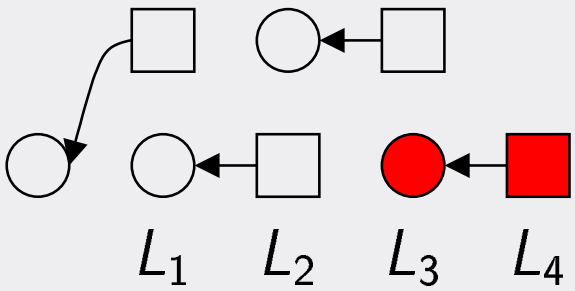


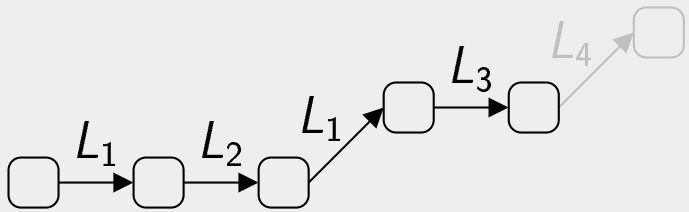


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

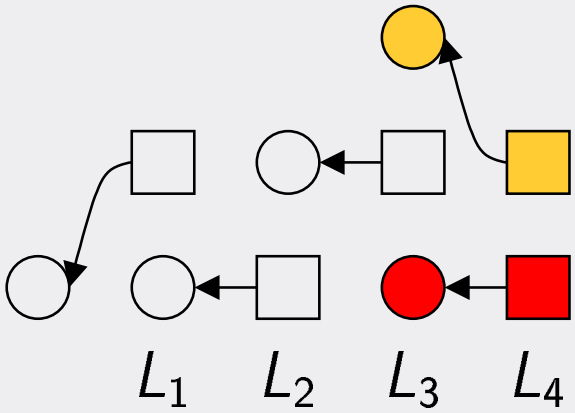


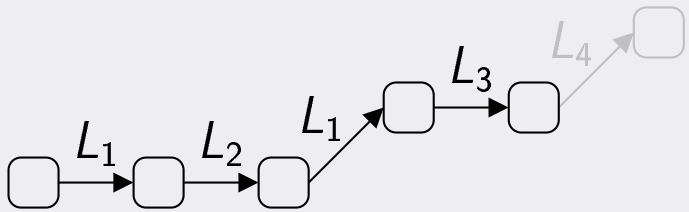


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

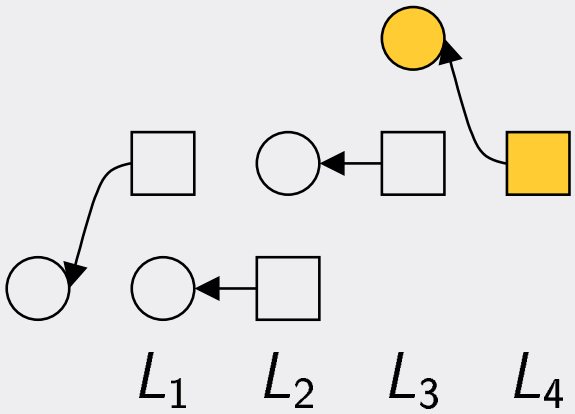


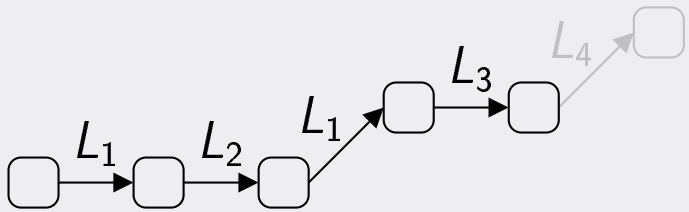


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

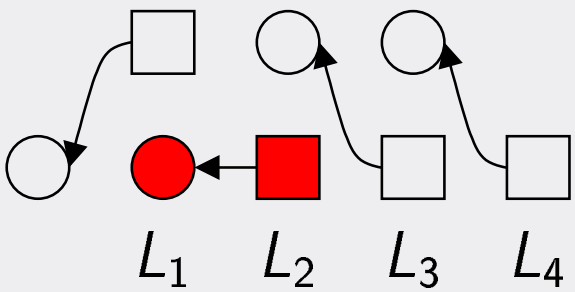


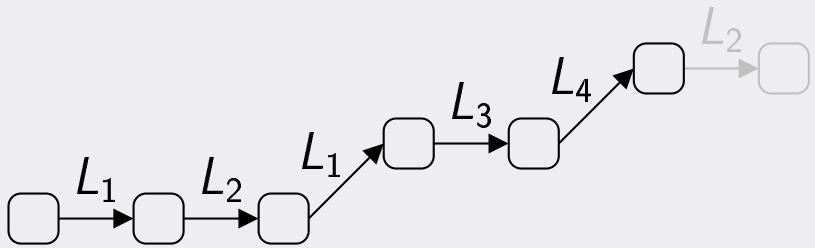


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)

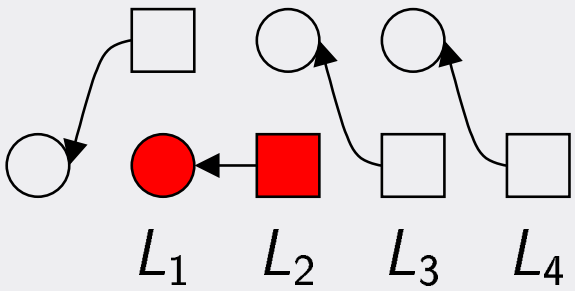


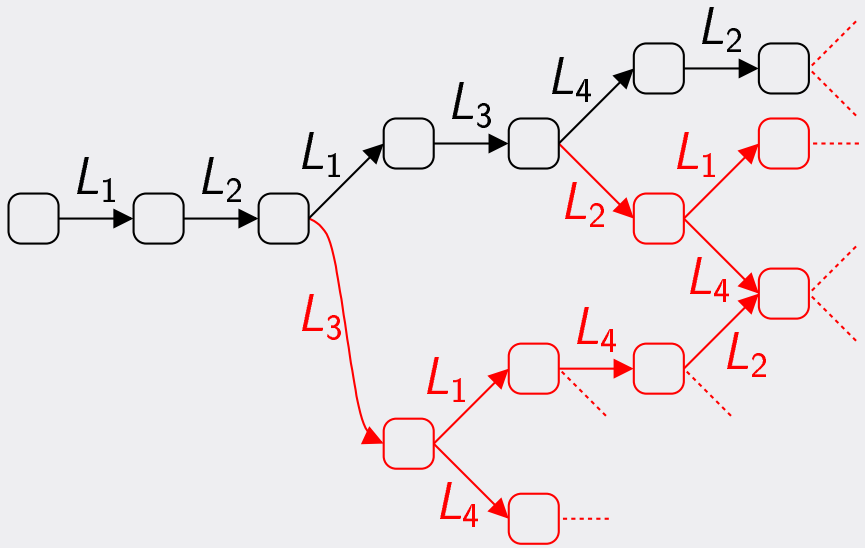


Modèle d'exécution des tâches itératives

Exemple d'évolution

1. Acquisition des requêtes (itération courante)
2. Poster de nouvelles requêtes (prochaine itération)
3. Calculer
4. Retirer les requêtes (itération courante)





Structure algébrique

Théorème

L'espace de configurations construit à partir d'une configuration initiale et de la règle d'évolution est un treillis combinatoire

Définition d'un treillis

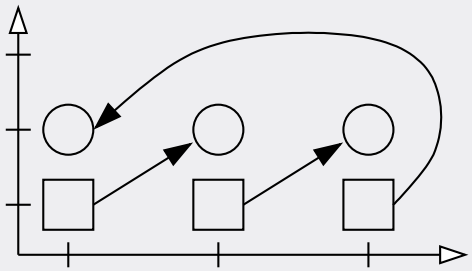
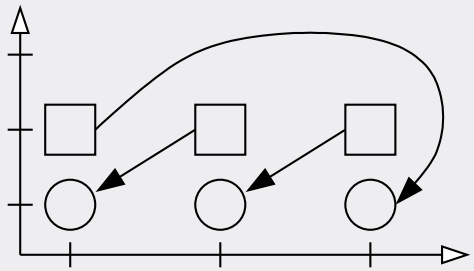
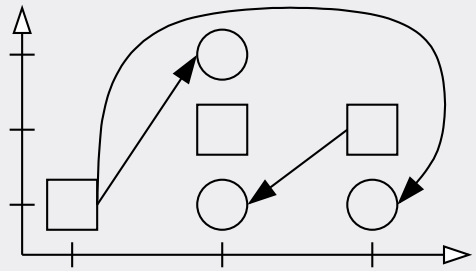
- Ensemble partiellement ordonné
- Tout couple d'éléments admet une borne supérieure et une borne inférieure uniques

Propriétés

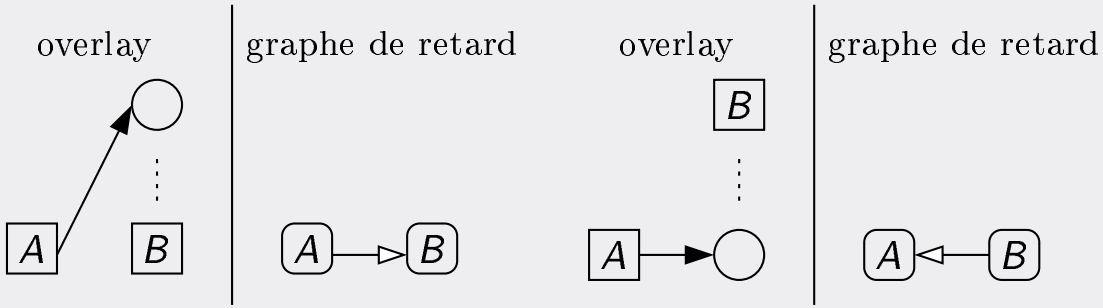
- Progression homogène des tâches
- Treillis fini $\Longleftrightarrow$ Présence d'un interblocage

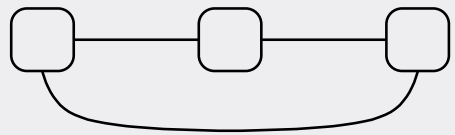
Interblocages

Détection des interblocages

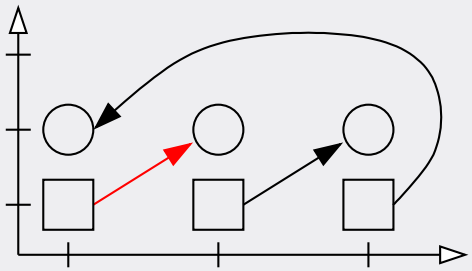
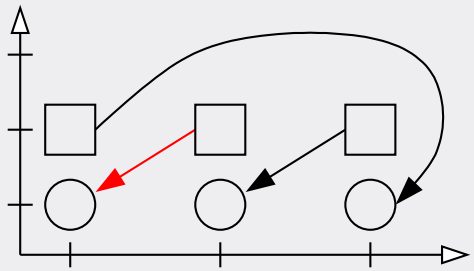
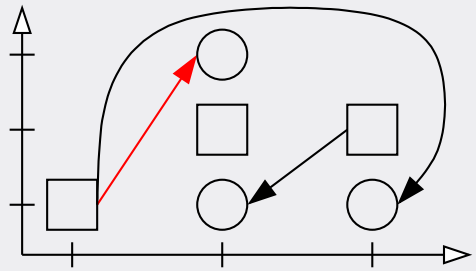
- Graphe orienté de retard : graphe de conflit orienté
- Interblocage $\Longleftrightarrow$ circuit dans le graphe de retard



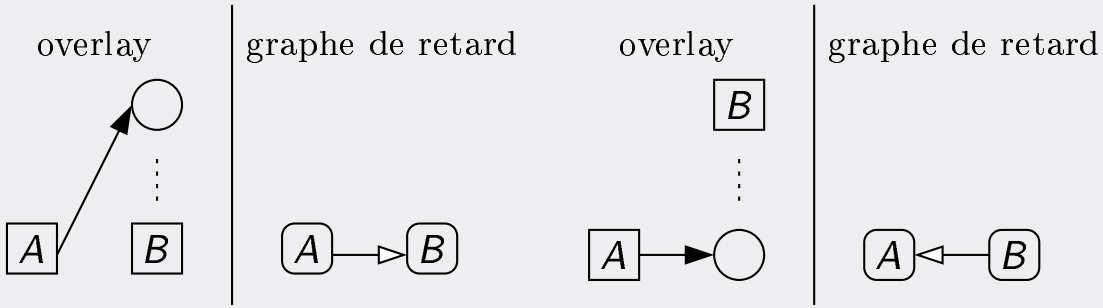


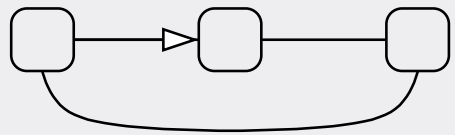
Interblocages

Détection des interblocages

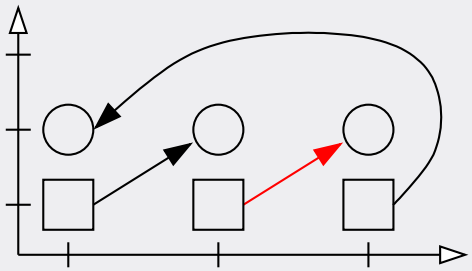
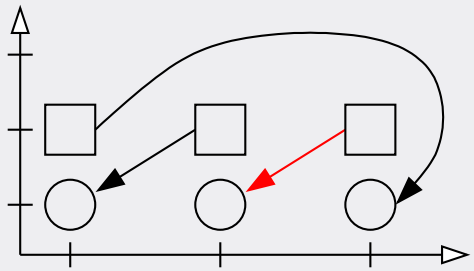
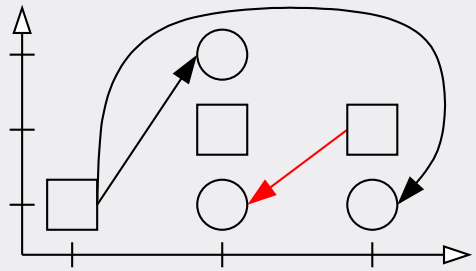
- Graphe orienté de retard : graphe de conflit orienté
- Interblocage $\Longleftrightarrow$ circuit dans le graphe de retard



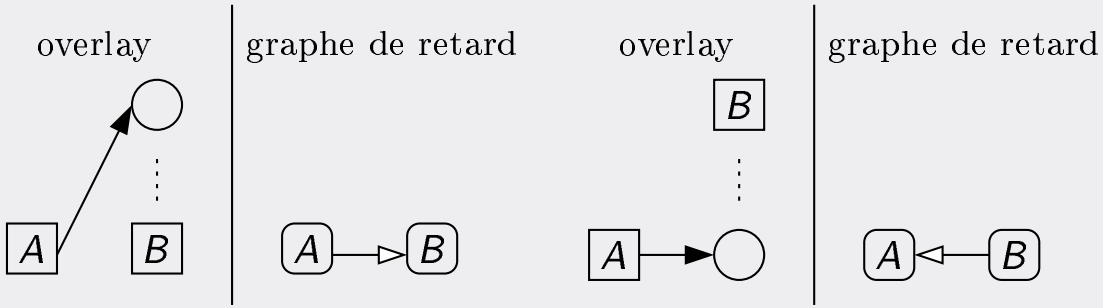


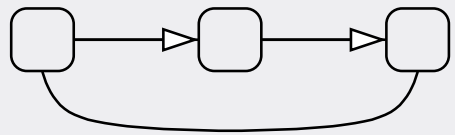
Interblocages

Détection des interblocages

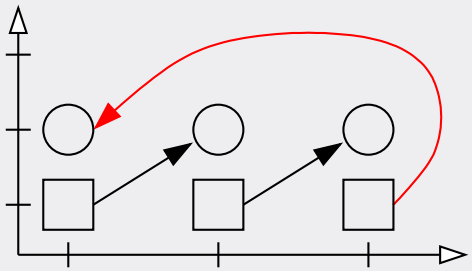
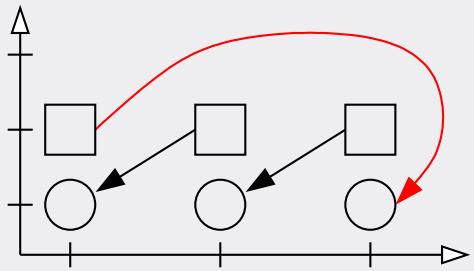
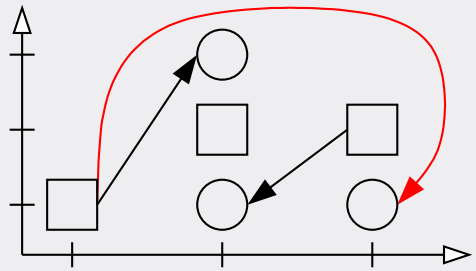
- Graphe orienté de retard : graphe de conflit orienté
- Interblocage $\Longleftrightarrow$ circuit dans le graphe de retard



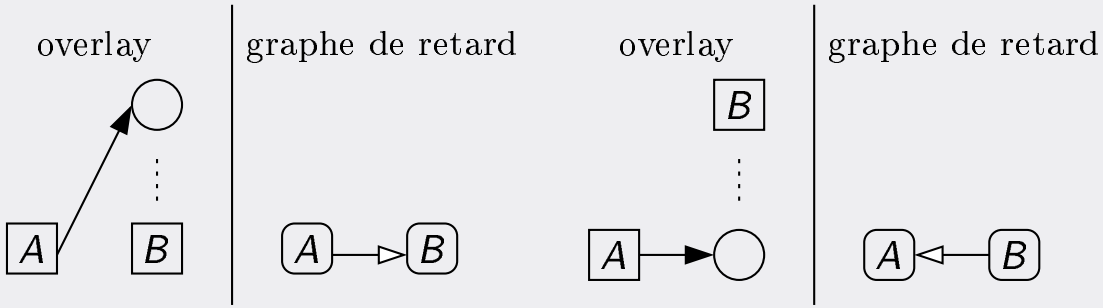


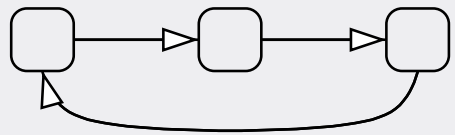
Interblocages

Détection des interblocages

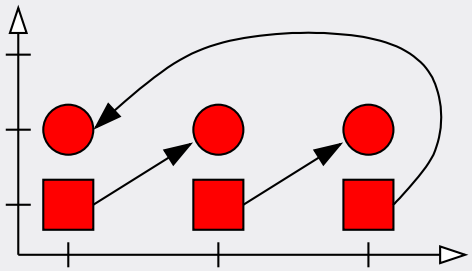
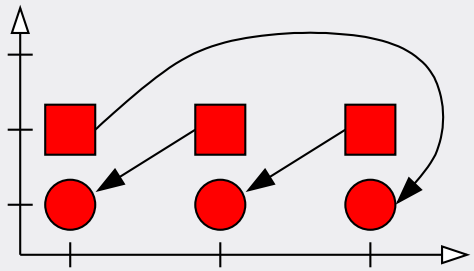
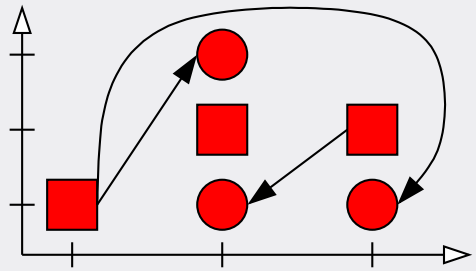
- Graphe orienté de retard : graphe de conflit orienté
- Interblocage $\Longleftrightarrow$ circuit dans le graphe de retard



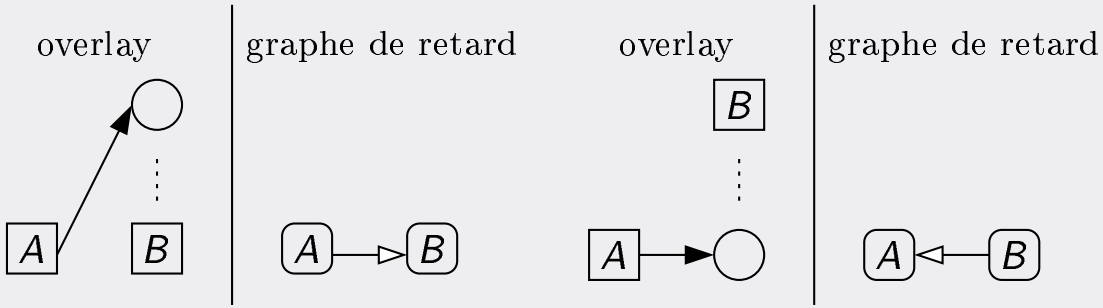


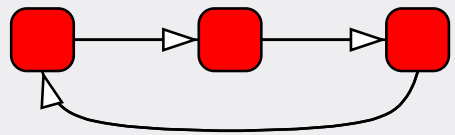
Interblocages

Détection des interblocages

- Graphe orienté de retard : graphe de conflit orienté
- Interblocage $\Longleftrightarrow$ circuit dans le graphe de retard





Propriétés du modèle sur les interblocages

Exécution avec ou sans interblocages

- Un interblocage ne peut pas être supprimé
- Un interblocage ne peut pas être créé
- Une exécution sans interblocage continue jusqu'à un évènement externe
  - Stabilisation
  - Action de l'utilisateur
- Les interblocages peuvent être détectés *a priori*

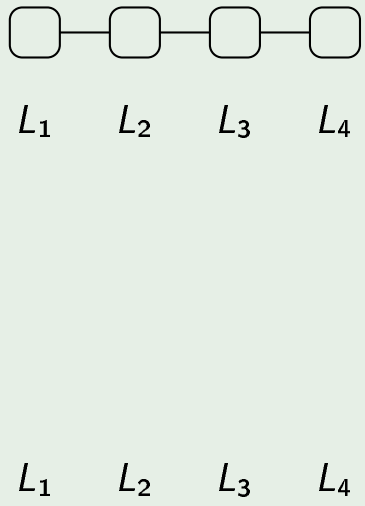
Plan

Initialisation d'un overlay

Algorithme d'initialisation

- Un ensemble de tâches $\mathcal{T}$ et un ensemble de lieux $\mathcal{L}$
- Pour chaque tâche $T_i \in \mathcal{T}$, une liste de lieux $(X_1, \ldots, X_{w_i})$ et une liste de lieux $(I_1, \ldots, I_{r_i})$ où placer des requêtes

1. Construire une représentation implicite du graphe de conflit $C(\mathcal{T})$
2. Calculer un coloriage $\mathcal{T}_1, \ldots, \mathcal{T}_n$ de $C(\mathcal{T})$
3.
```
pour tout lieu L ∈ L faire
    p(L) ← 0
fin pour
pour toute couleur c de 1 à n faire
    pour toute tâche T ∈ T_c faire en parallèle
        pour toute requête exclusive X de T sur le
        lieu X_i faire
            p(X_i) ← p(X_i) + 1
            priorité de X ← p(X_i)
            p(X_i) ← p(X_i) + 1
        fin pour
        pour toute requête inclusive I de T sur le lieu
        I_i faire
            priorité de I ← p(I_i)
        fin pour
    fin pour
fin pour
```

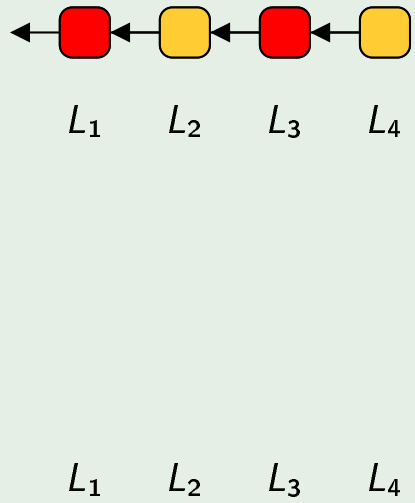


Initialisation d'un overlay

Algorithme d'initialisation

- Un ensemble de tâches $\mathcal{T}$ et un ensemble de lieux $\mathcal{L}$
- Pour chaque tâche $T_i \in \mathcal{T}$, une liste de lieux $(X_1, \ldots, X_{w_i})$ et une liste de lieux $(l_1, \ldots, l_{r_i})$ où placer des requêtes

1. Construire une représentation implicite du graphe de conflit $C(\mathcal{T})$
2. Calculer un coloriage $\mathcal{T}_1, \ldots, \mathcal{T}_n$ de $C(\mathcal{T})$
3.
```
pour tout lieu L ∈ L faire
    p(L) ← 0
fin pour
pour toute couleur c de 1 à n faire
    pour toute tâche T ∈ T_c faire en parallèle
        pour toute requête exclusive X de T sur le
        lieu X_i faire
            p(X_i) ← p(X_i) + 1
            priorité de X ← p(X_i)
            p(X_i) ← p(X_i) + 1
        fin pour
        pour toute requête inclusive l de T sur le lieu
        l_i faire
            priorité de l ← p(l_i)
        fin pour
    fin pour
fin pour
```

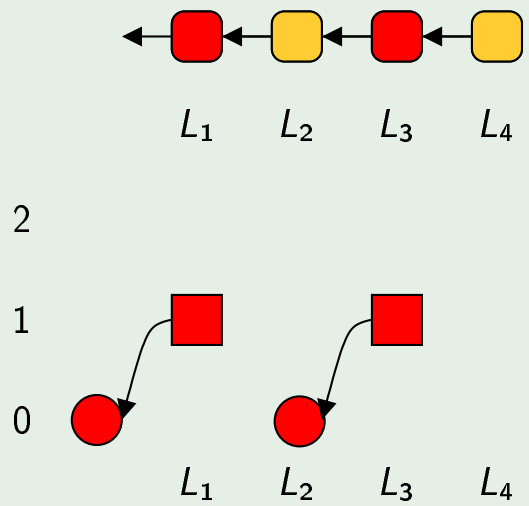


Initialisation d'un overlay

Algorithme d'initialisation

- Un ensemble de tâches $\mathcal{T}$ et un ensemble de lieux $\mathcal{L}$
- Pour chaque tâche $T_i \in \mathcal{T}$, une liste de lieux $(X_1, \ldots, X_{w_i})$ et une liste de lieux $(l_1, \ldots, l_{r_i})$ où placer des requêtes

1. Construire une représentation implicite du graphe de conflit $C(\mathcal{T})$
2. Calculer un coloriage $\mathcal{T}_1, \ldots, \mathcal{T}_n$ de $C(\mathcal{T})$
3.
```
pour tout lieu L ∈ ℒ faire
    p(L) ← 0
fin pour
pour toute couleur c de 1 à n faire
    pour toute tâche T ∈ T_c faire en parallèle
        pour toute requête exclusive X de T sur le
        lieu X_i faire
            p(X_i) ← p(X_i) + 1
            priorité de X ← p(X_i)
            p(X_i) ← p(X_i) + 1
        fin pour
        pour toute requête inclusive l de T sur le lieu
        l_i faire
            priorité de l ← p(l_i)
        fin pour
    fin pour
fin pour
```

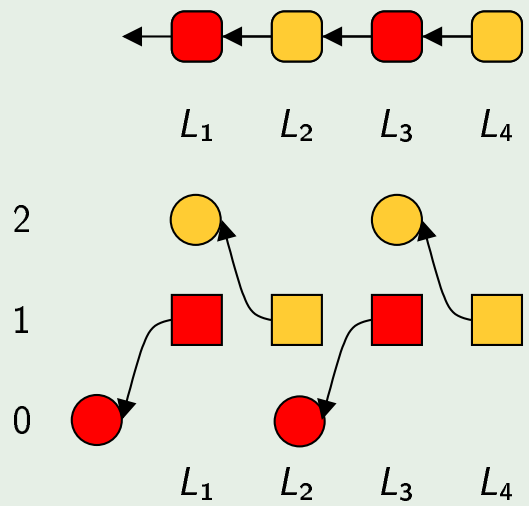


Initialisation d'un overlay

Algorithme d'initialisation

- Un ensemble de tâches $\mathcal{T}$ et un ensemble de lieux $\mathcal{L}$
- Pour chaque tâche $T_i \in \mathcal{T}$, une liste de lieux $(X_1, \ldots, X_{w_i})$ et une liste de lieux $(l_1, \ldots, l_{r_i})$ où placer des requêtes

1. Construire une représentation implicite du graphe de conflit $C(\mathcal{T})$
2. Calculer un coloriage $\mathcal{T}_1, \ldots, \mathcal{T}_n$ de $C(\mathcal{T})$
3.
```
pour tout lieu L ∈ ℒ faire
    p(L) ← 0
fin pour
pour toute couleur c de 1 à n faire
    pour toute tâche T ∈ T_c faire en parallèle
        pour toute requête exclusive X de T sur le
        lieu X_i faire
            p(X_i) ← p(X_i) + 1
            priorité de X ← p(X_i)
            p(X_i) ← p(X_i) + 1
        fin pour
        pour toute requête inclusive l de T sur le lieu
        l_i faire
            priorité de l ← p(l_i)
        fin pour
    fin pour
fin pour
```

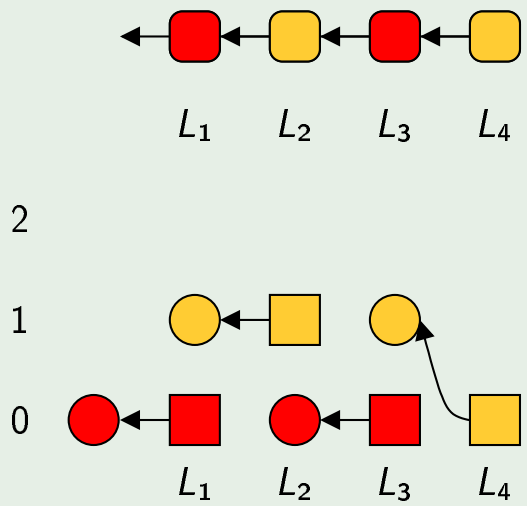


Initialisation d'un overlay

Algorithme d'initialisation

- Un ensemble de tâches $\mathcal{T}$ et un ensemble de lieux $\mathcal{L}$
- Pour chaque tâche $T_i \in \mathcal{T}$, une liste de lieux $(X_1, \ldots, X_{w_i})$ et une liste de lieux $(l_1, \ldots, l_{r_i})$ où placer des requêtes

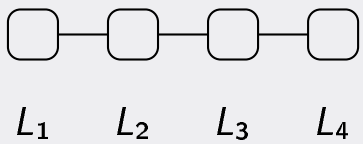
1. Construire une représentation implicite du graphe de conflit $C(\mathcal{T})$
2. Calculer un coloriage $\mathcal{T}_1, \ldots, \mathcal{T}_n$ de $C(\mathcal{T})$
3.
```
pour tout lieu L ∈ ℒ faire
    p(L) ← 0
fin pour
pour toute couleur c de 1 à n faire
    pour toute tâche T ∈ T_c faire en parallèle
        pour toute requête exclusive X de T sur le
        lieu X_i faire
            p(X_i) ← p(X_i) + 1
            priorité de X ← p(X_i)
            p(X_i) ← p(X_i) + 1
        fin pour
        pour toute requête inclusive l de T sur le lieu
        l_i faire
            priorité de l ← p(l_i)
        fin pour
    fin pour
fin pour
```



Propriétés de l'initialisation

Impact du coloriage

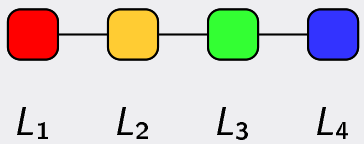
- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations



Propriétés de l'initialisation

Impact du coloriage

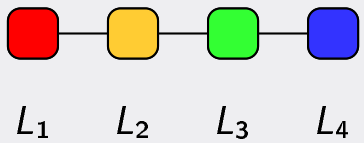
- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

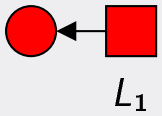


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

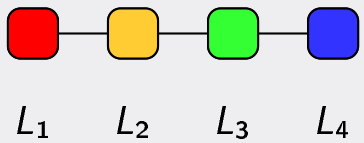


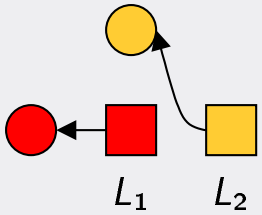


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

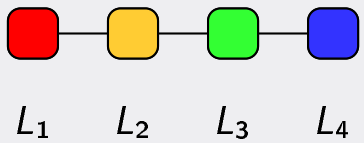


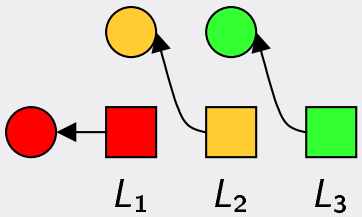


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

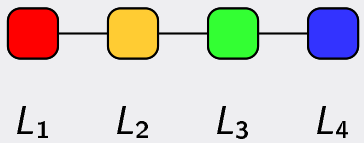


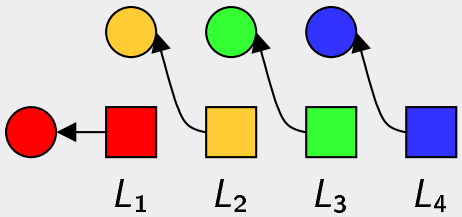


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

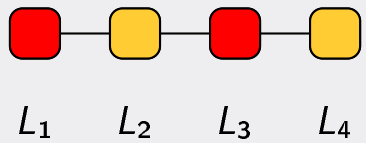


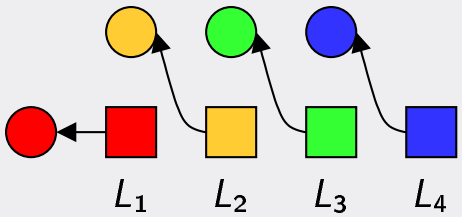


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

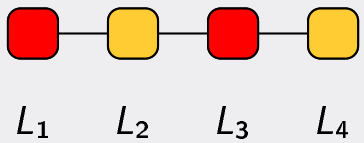


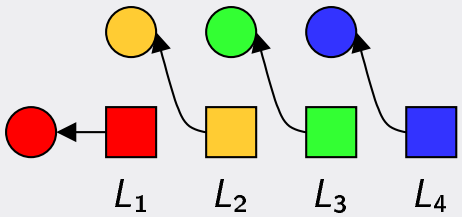


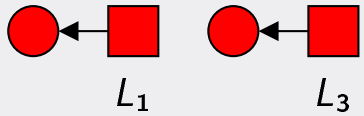
Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations



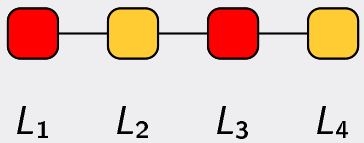


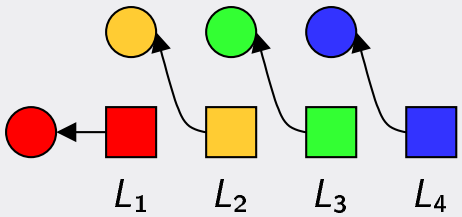


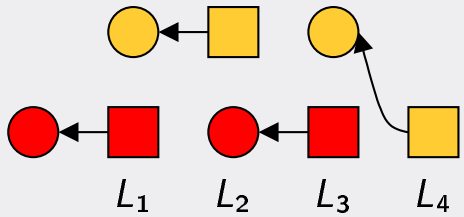
Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations



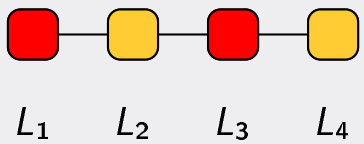


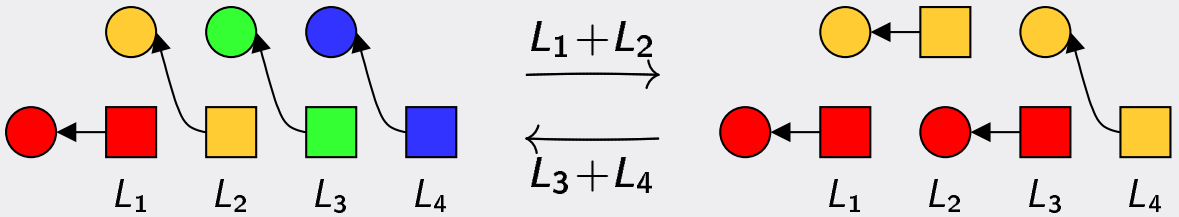


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de la chaîne linéaire de 4 tâches
- Bénin : même espace de configurations

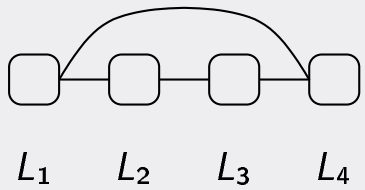




Propriétés de l'initialisation

Impact du coloriage

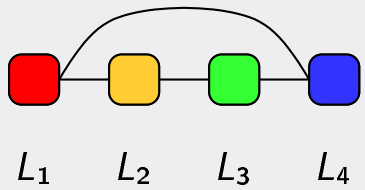
- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints



Propriétés de l'initialisation

Impact du coloriage

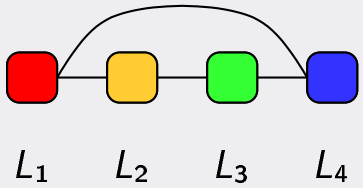
- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints



Propriétés de l'initialisation

Impact du coloriage

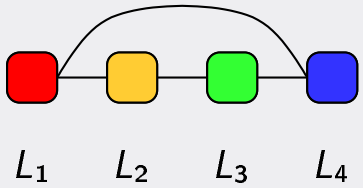
- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints

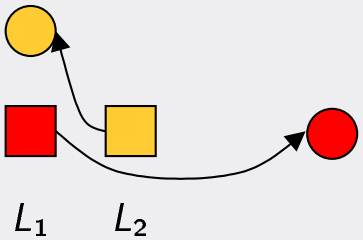


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints

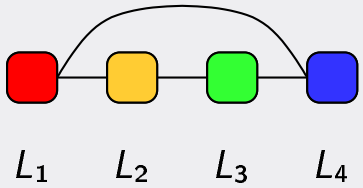


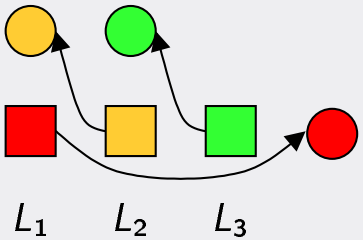


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints

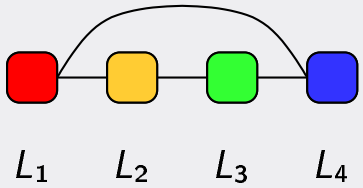


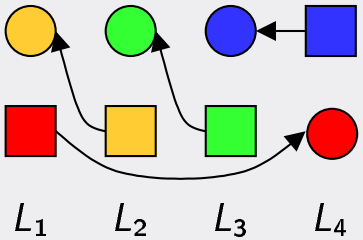


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints

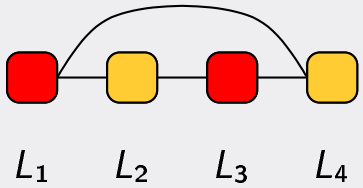


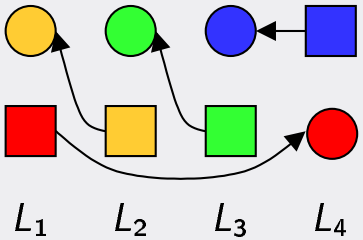


Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints

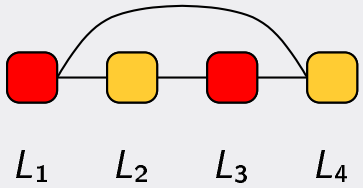


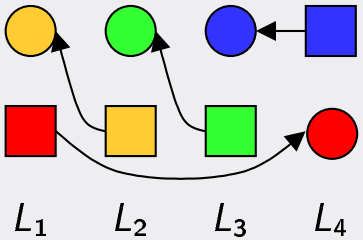


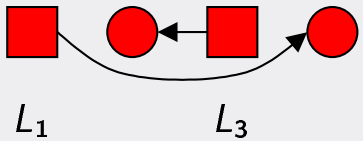
Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints



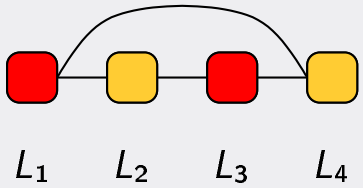


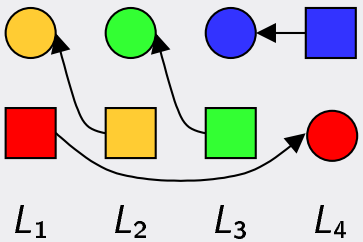


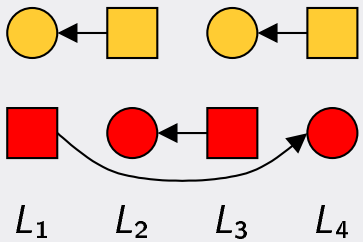
Propriétés de l'initialisation

Impact du coloriage

- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints



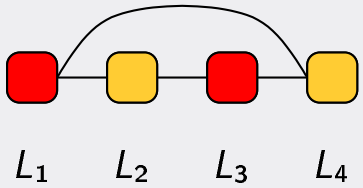


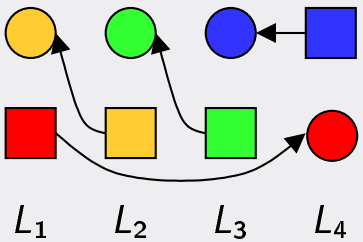


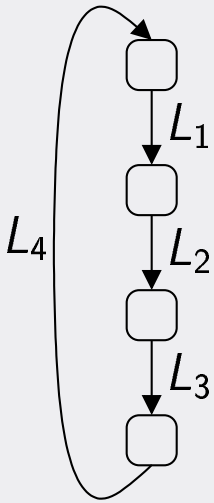
Propriétés de l'initialisation

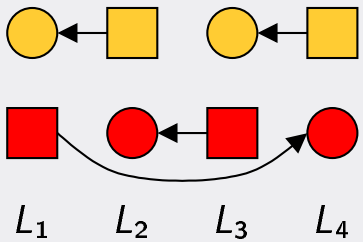
Impact du coloriage

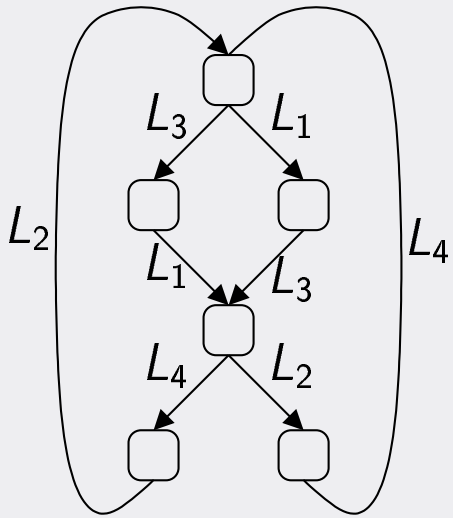
- Deux coloriages de l'anneau chaîné de 4 tâches
- Espaces de configurations disjoints











Plan

Détails d'implémentation

parXXL

- C++
- *Livermore Kernel 23* pour le stockage par blocs
- Réutilisation maximale du code pour le stockage étendu
  - Transformation vers l'agencement classique
  - Réutilisation du code de calcul
  - Transformation vers l'agencement étendu
- E/S avec **read** et **write** au lieu de **mmap**
- ORWL basés sur des **pthread_cond_t**

Plan

Grid'5000

Machines de 3 clusters

- *Grelon* - Nancy
  - Bi-processeurs bi-cœurs
  - 2 Go RAM
- *Capricorne* - Lyon
  - Bi-processeurs mono-cœurs
  - 2 Go RAM
- *Chinqchint* - Lille
  - Bi-processeurs quad-cœurs
  - 8 Go RAM

Plan

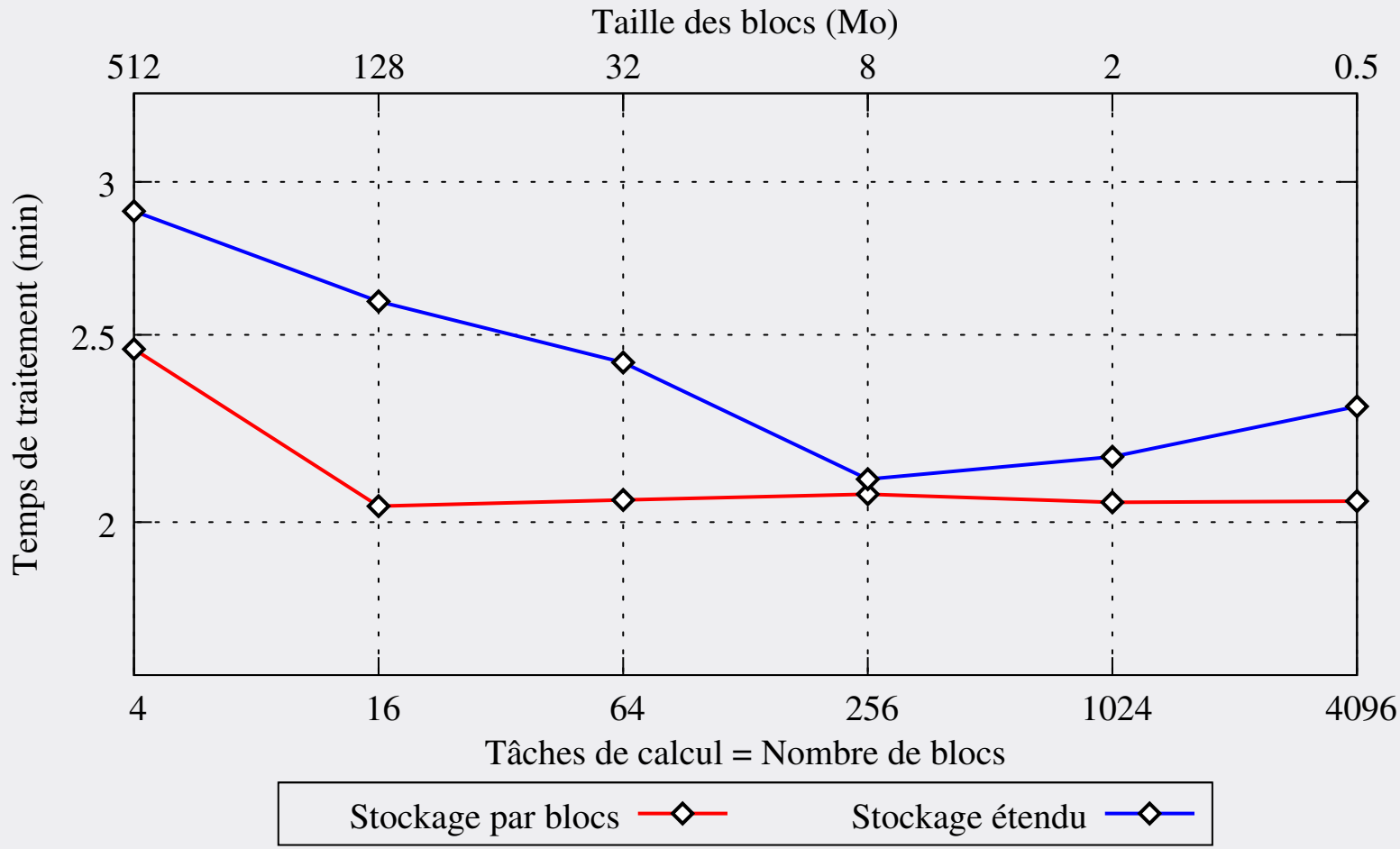
Stockage étendu

Expériences

- Comparaison avec le stockage par blocs
- Comparaison des coûts de pré- et post-traitements
- Comparaison des temps d'exécution d'une itération du *Livermore Kernel 23*

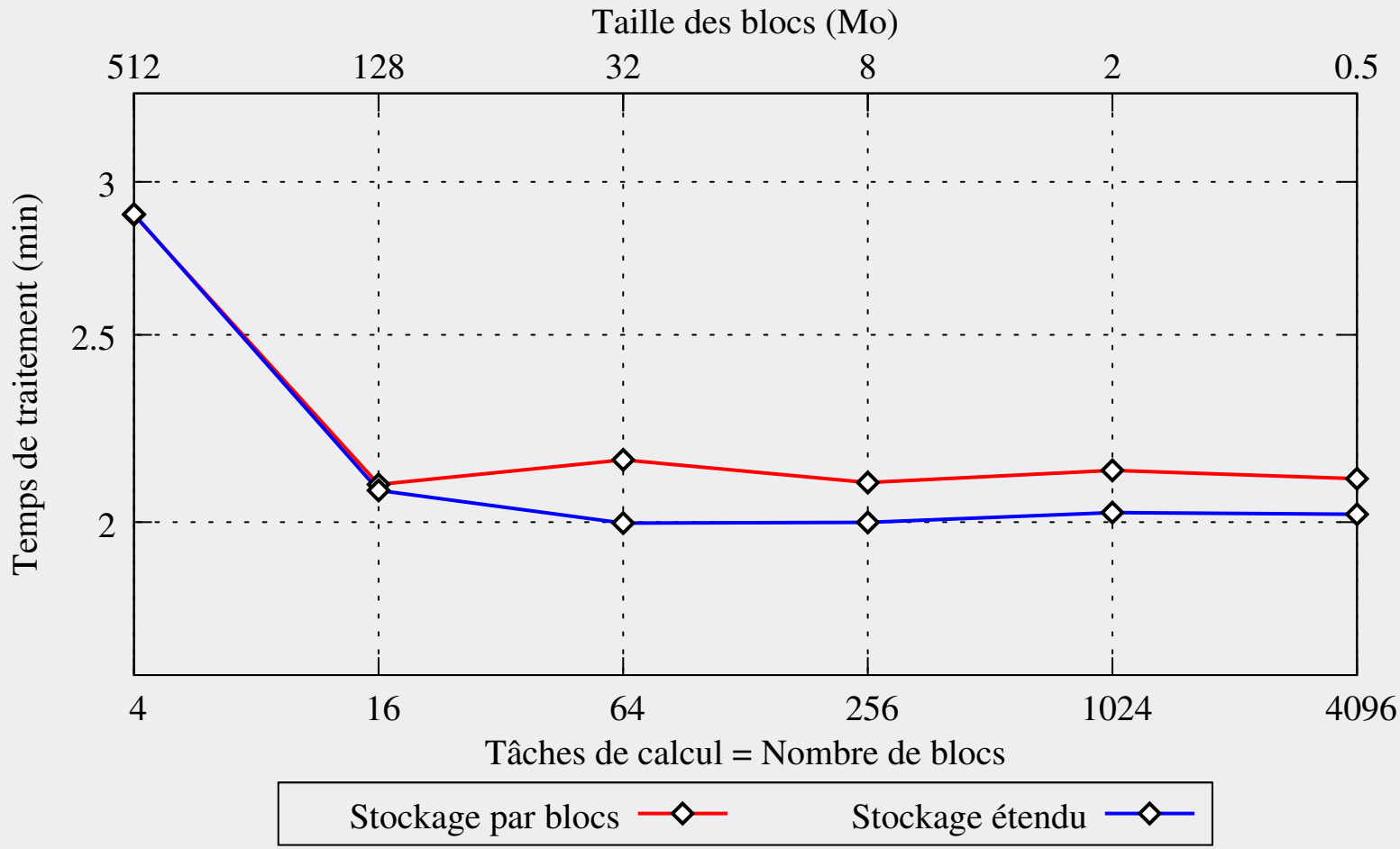
Coûts de pré-traitement

Sur *Chinqchint*



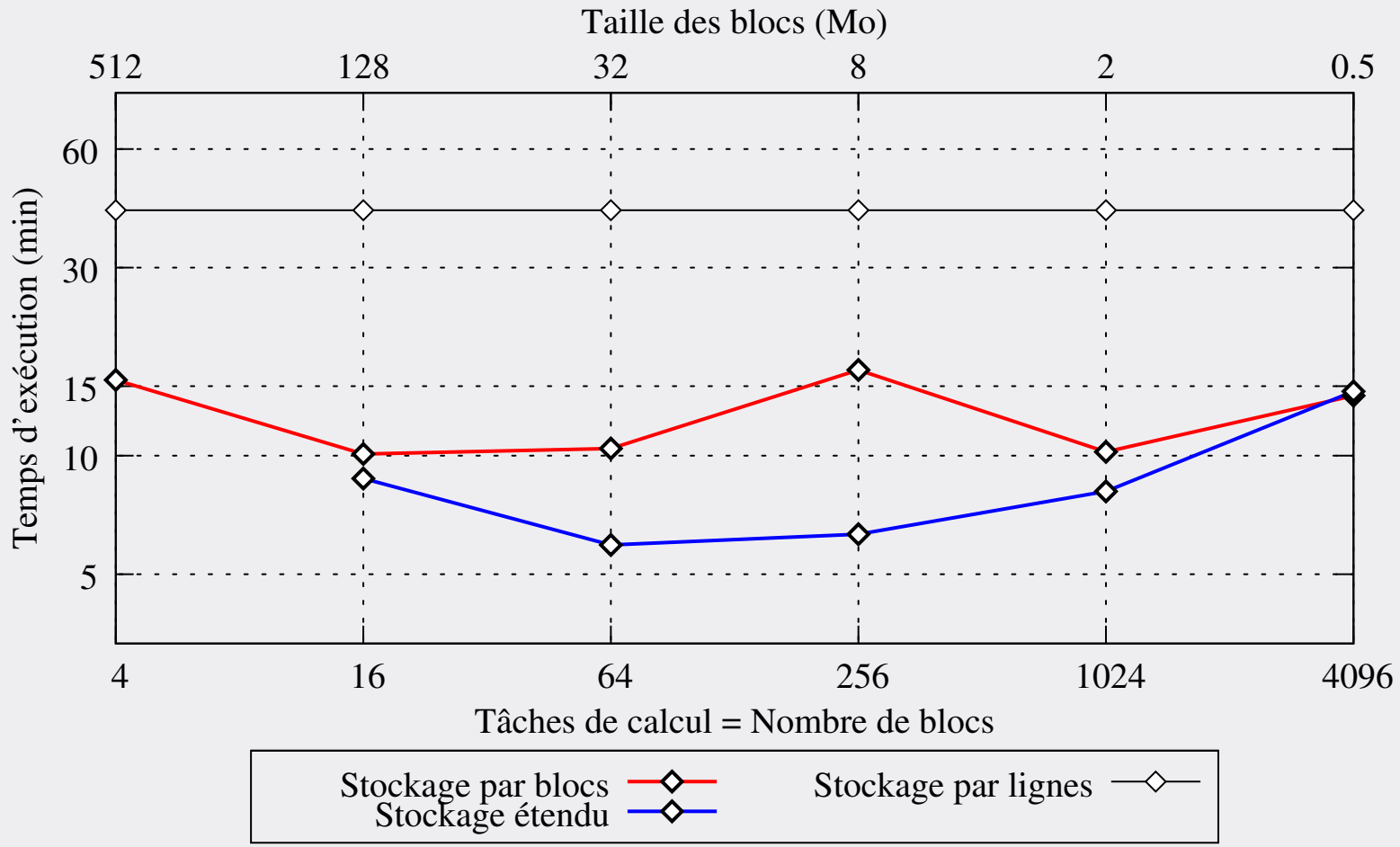
Coûts de post-traitement

Sur *Chinqchint*



Comparaison des performances

Sur *Capricorne*



Comparaison des performances

Une itération du *Livermore Kernel 23*

	Gain (%)		
	$\frac{\text{Blocs}}{\text{Lignes}}$	$\frac{\text{Étendu}}{\text{Lignes}}$	$\frac{\text{Étendu}}{\text{Blocs}}$
Grelon	74.8	89.8	59.6
Capricorne	75.9	85.5	39.6
Chinqchint	73.5	75.3	6.7

- Le stockage étendu est plus performant que le stockage par blocs

Out-of-Core

Expérience

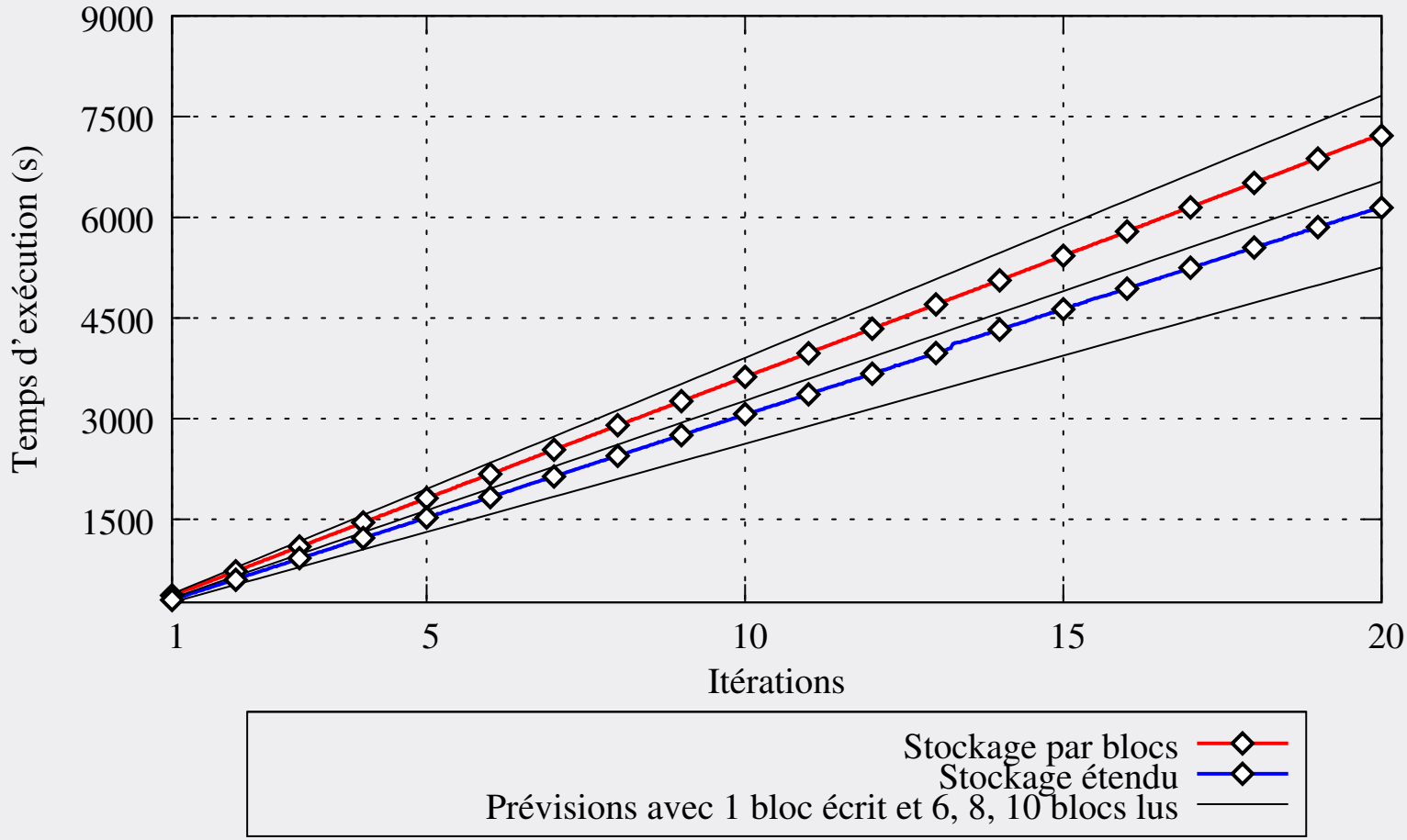
- 20 itérations du *Livermore Kernel 23*

Prévisions

- Pire cas : lecture de 10 blocs
- Meilleur cas : lecture de 6 blocs
- Dans tous les cas : écriture d'1 bloc

Comparaison des performances

Sur *Grelon*



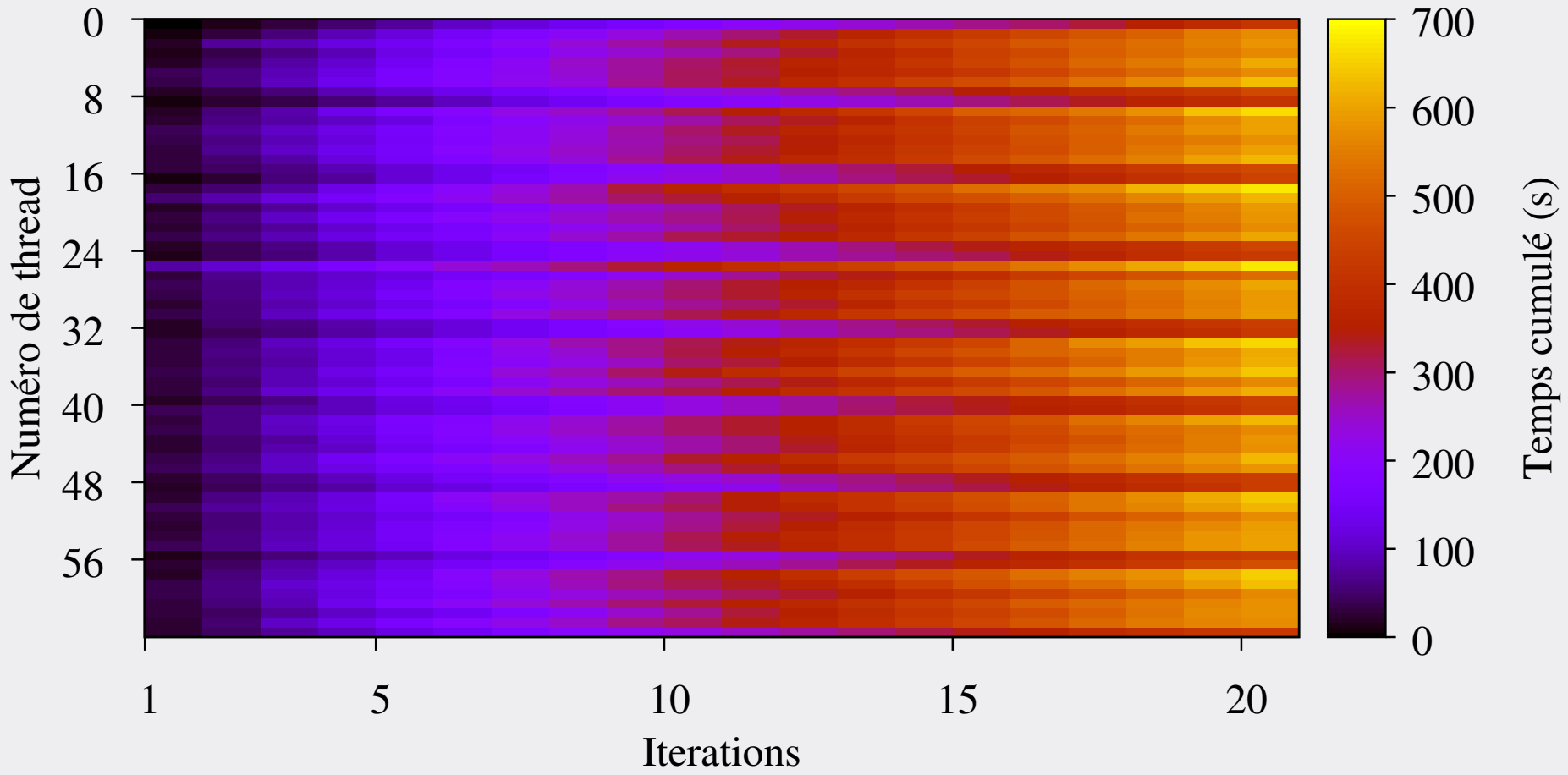
Homogénéité

Expérience

- 20 itérations du *Livermore Kernel 23*
- 2 mesures par thread et par itération
  - Temps de calcul et d'E/S
  - Temps de synchronisation sur les ORWL

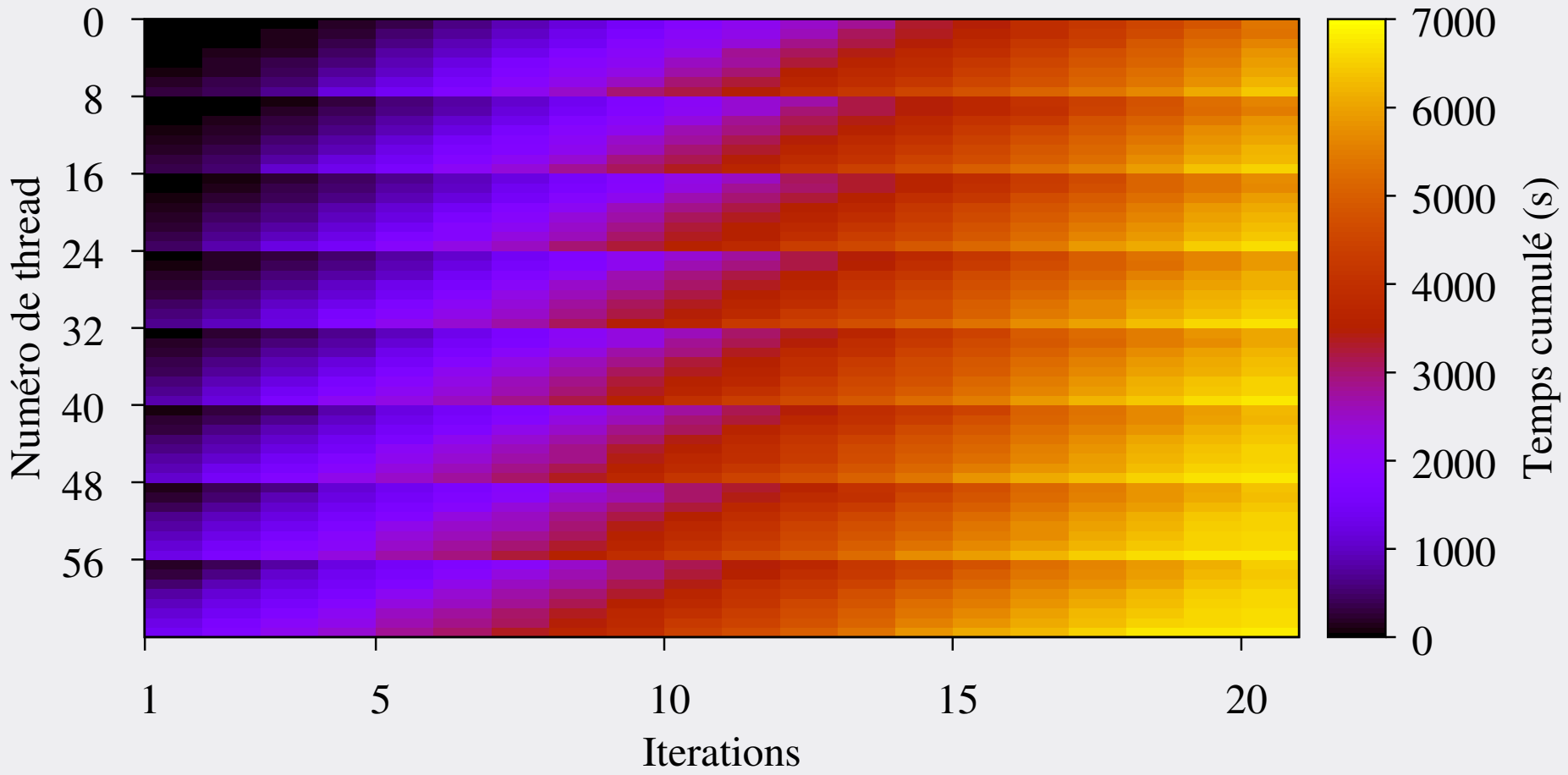
Temps de calcul et d'E/S

Sur *Grelon*



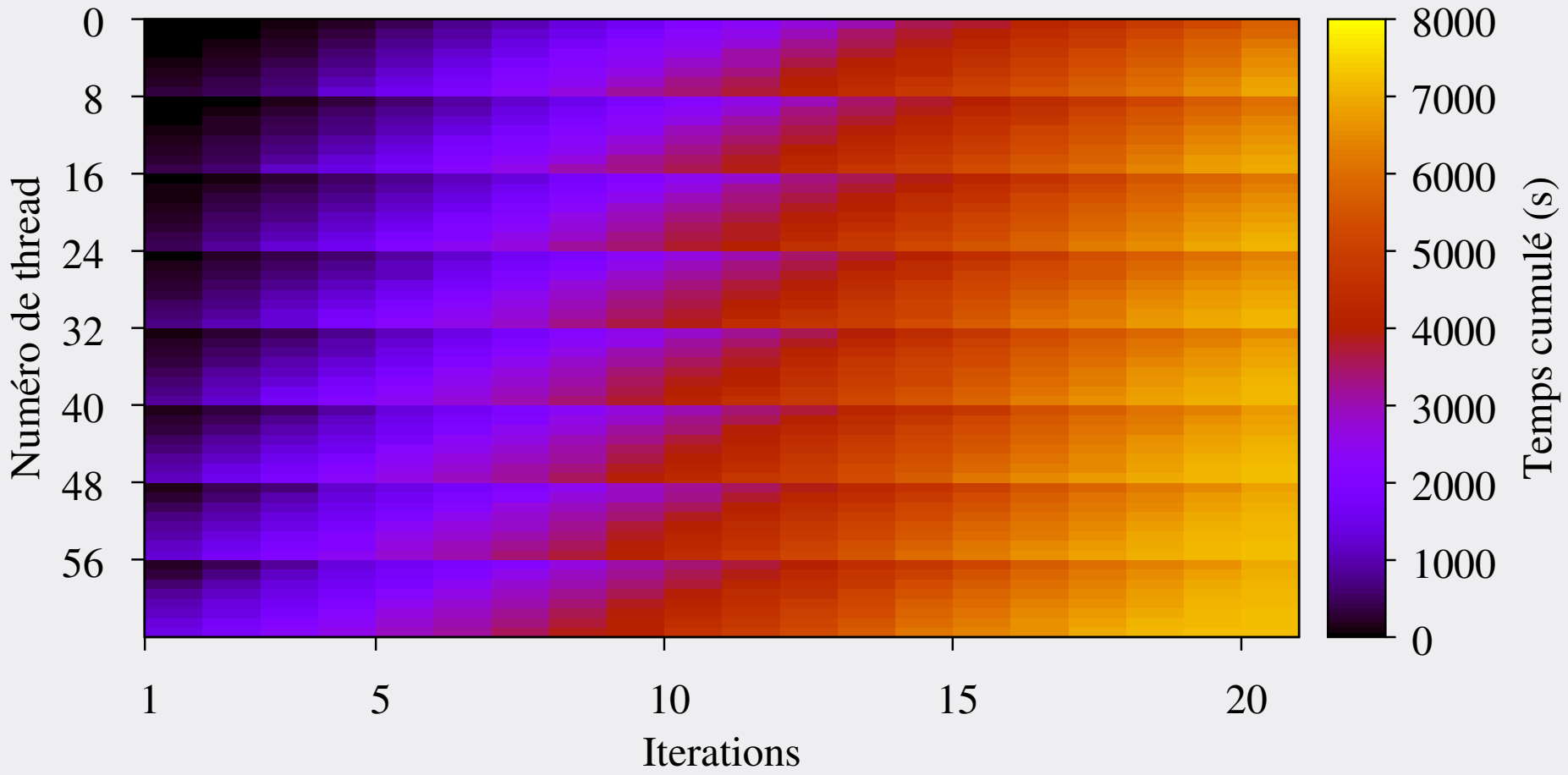
Temps de synchronisation

Sur *Grelon*



Temps total

Sur *Grelon*



Plan

Conclusions

Stockage étendu

- Gain substantiel par rapport au stockage par blocs
- Facilité de mise en œuvre
- Surcoût négligeable

ORWL

- Absence d'interblocage
- Homogénéité
- Interface similaire aux outils existants
- Validation expérimentale des performances

Perspectives

Applications réelles

- Collaboration avec D. Komatitsch et R. Martin de l'université de Pau
- Simulation de transferts thermiques
  - Version séquentielle en Fortran
  - Portage en C++ au cours d'un stage
  - En cours de parallélisation
- Simulation de propagation d'une onde sismique
  - Problème tri-dimensionnel

Nouvelles utilisations

- Extension du modèle des ORWL
- Implémentation distribuée
- Modélisation des verrouillages d'intervalles d'octets

Publications

- Pierre-Nicolas Clauss, Jens Gustedt et Frédéric Suter.
  Out-of-Core Wavefront Computations with Reduced Synchronization.
  16th Euromicro International Conference on Parallel, Distributed and network-based Processing.
  IEEE, 2008.
- Pierre-Nicolas Clauss et Jens Gustedt.
  Iterative Computations with Ordered Read-Write Locks.
  Journal of Parallel and Distributed Computing
  Elsevier, 2009.
- Pierre-Nicolas Clauss et Jens Gustedt.
  Experimenting Iterative Computations with Ordered Read-Write Locks.
  18th Euromicro International Conference on Parallel, Distributed and network-based Processing.
  IEEE, 2010.