Ordonnancement d'applications de type stencil sur cluster hybrides CPU/GPU

Université de Bordeaux - LaBRI - INRIA Bordeaux Sud-Ouest - Équipe STORM

Loris Lucido

26 septembre 2016

Domaine du Calcul Haute Performance

- Résoudre un problème
  - de plus en plus rapidement
  - de taille de plus en plus grande
- Application de simulation en :
  - climatologie
  - aérospatiale
  - dynamique moléculaire

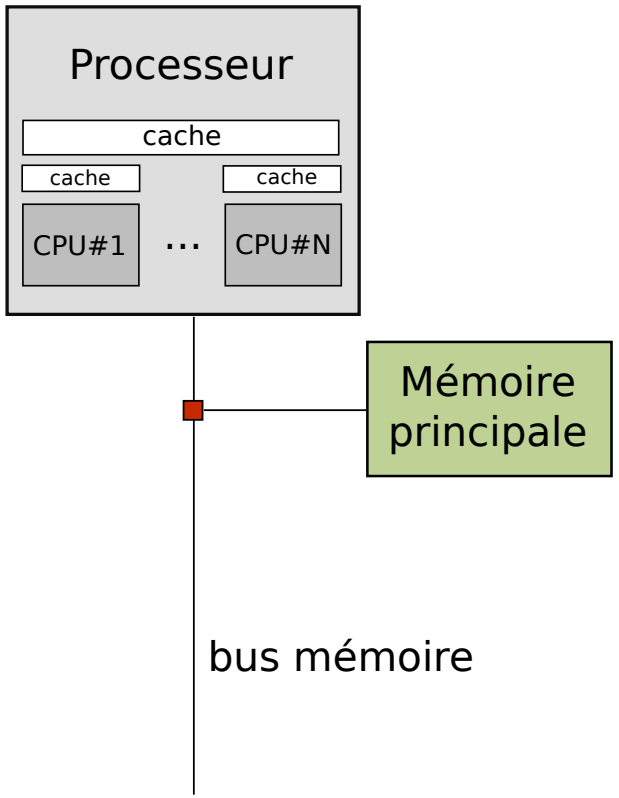
Présentation de l'équipe STORM

STatic Optimizations Runtime Methods, trois axes de recherche :

- Langage de haut niveau spécifique à un domaine (*e.g. QIRAL*)
- Outils d'analyse de performance et d'aide à l'optimisation (*e.g. MAQAO*)
- Support d'exécution pour des calculateurs hétérogènes (*StarPU*)

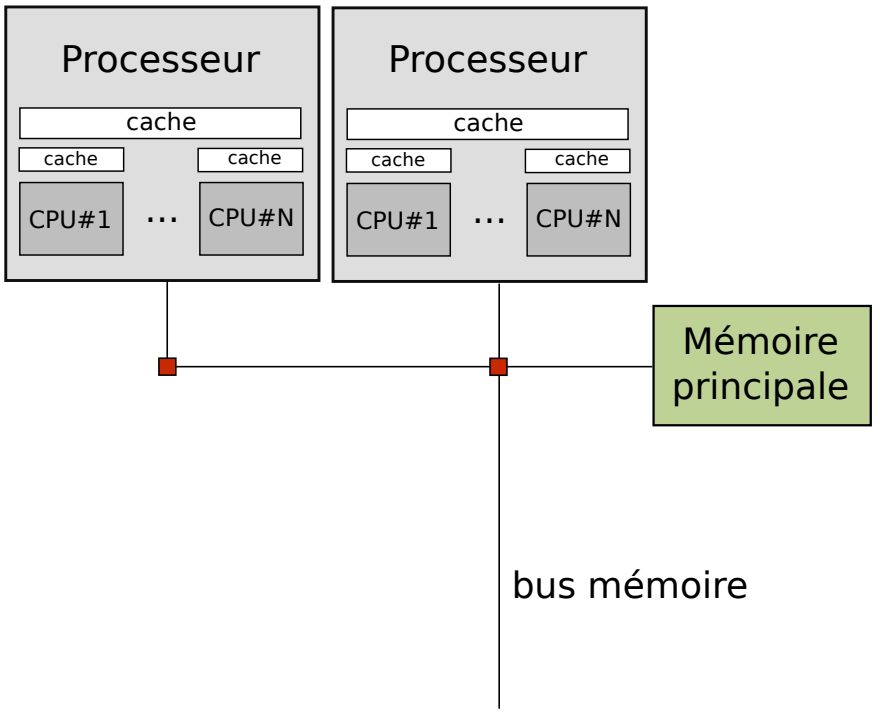
Architecture hétérogène

- Des dizaines de cœurs par processeurs



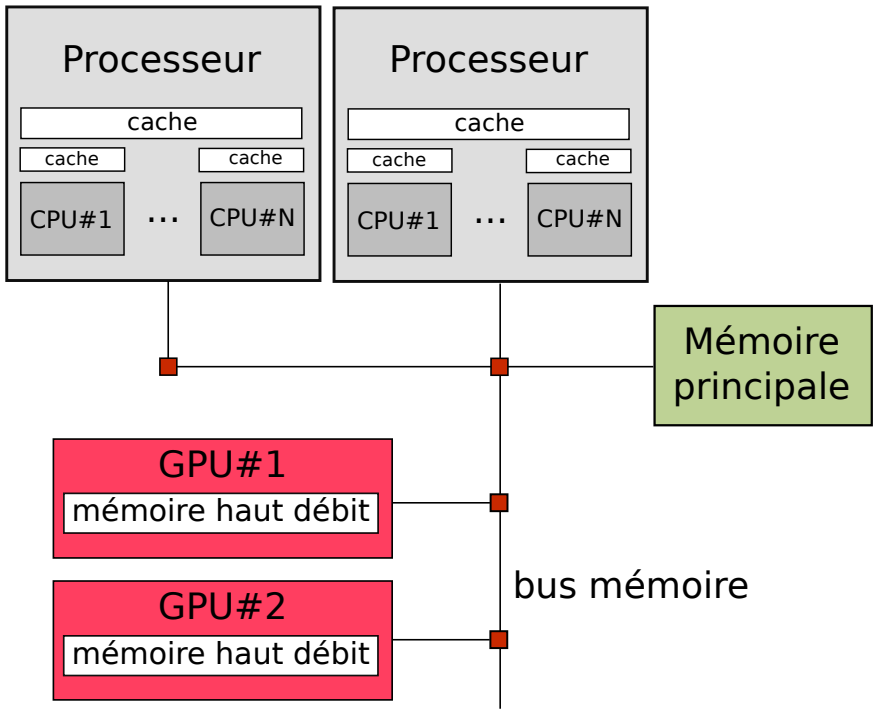
Architecture hétérogène

- Des dizaines de cœurs par processeurs
- Plusieurs processeurs par machine



Architecture hétérogène

- Des dizaines de cœurs par processeurs
- Plusieurs processeurs par machine
- Plusieurs cartes graphiques dédiées au calcul :
  - parallélisme massif



Architecture hétérogène

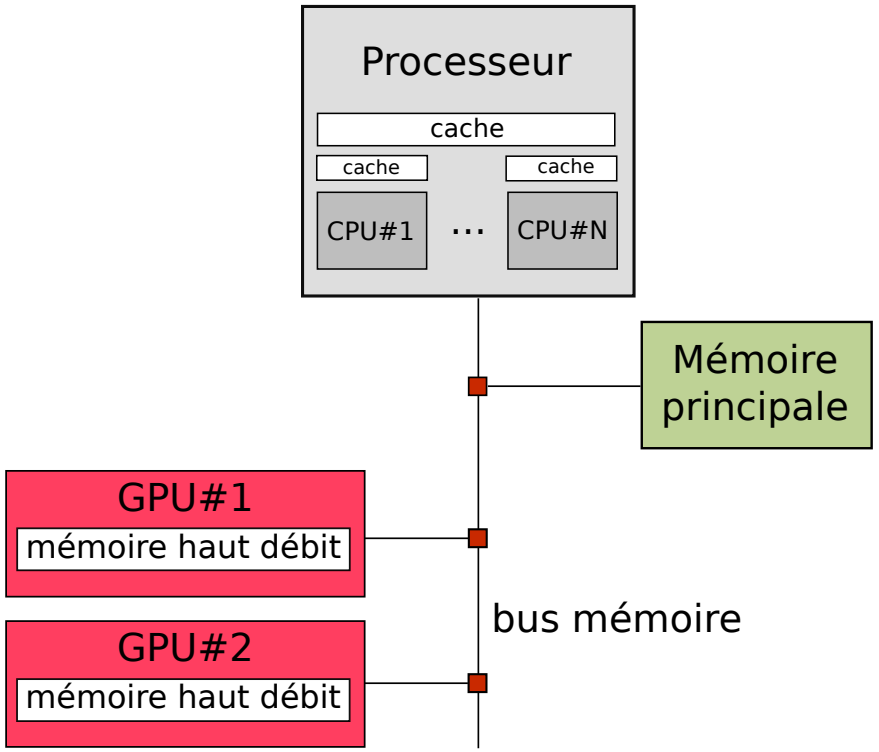


Figure: Topologie mémoire d'une machine avec deux cartes graphiques.

Carte graphique dédiée au calcul :

Architecture hétérogène

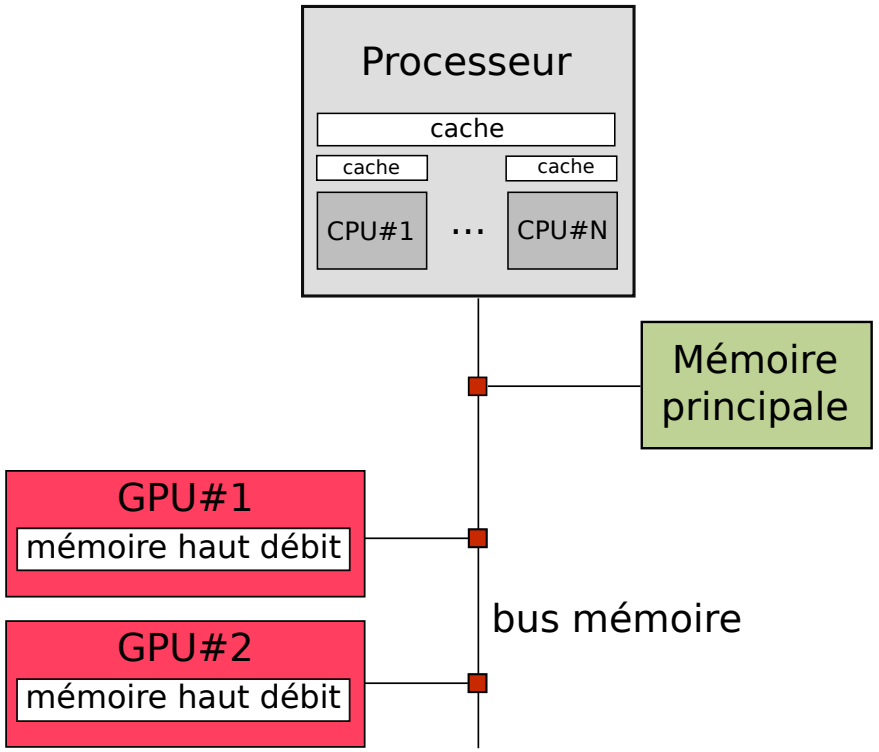


Figure: Topologie mémoire d'une machine avec deux cartes graphiques.

Carte graphique dédiée au calcul :

- Mémoire dédiée en quantité limitée

Architecture hétérogène

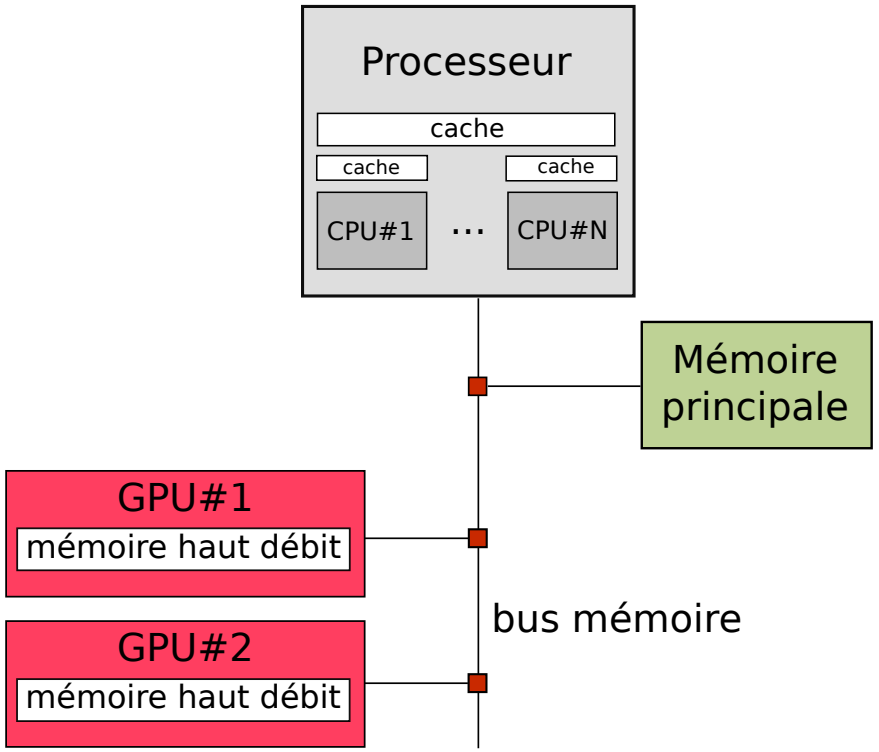


Figure: Topologie mémoire d'une machine avec deux cartes graphiques.

Carte graphique dédiée au calcul :

- Mémoire dédiée en quantité limitée
- Pas d'accès direct à la mémoire principale

Le support d'exécution (ou *Runtime*) StarPU

- Entre le système d'exploitation et l'application
- Exploite une architecture :
  - portable et générique
  - paradigme de programmation parallèle en tâches
- Stratégies d'ordonnancement (ordonnanceur)

Programmation parallèle en tâches

- Découpage du problème en sous-ensembles : tâches
- Exécution parallèle des tâches sur différentes unités de calcul
- Dépendances de données entre les tâches

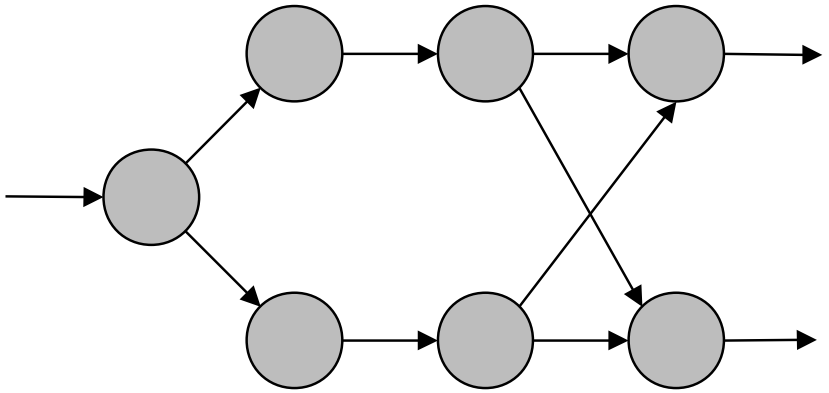


Figure: Graphe de dépendances de tâches.

Ordonnancement des tâches par le support d'exécution

- Ordonnancement du graphe de tâches sur les différents unités de calcul ou ouvriers :
  - cœurs CPU (processeur)
  - GPU (carte graphique)
- Choisir le «meilleur ouvrier» pour une tâche donnée
- Transférer les résultats des calculs

Applications stencils

Qu'est-ce qu'une application stencil ?

- Stencil : motif ou pochoir que l'on applique par répétition sur l'ensemble du domaine étudié

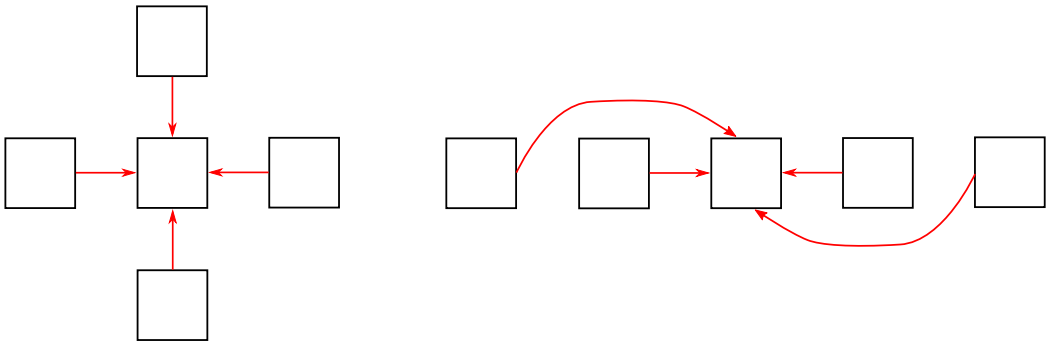


Figure: Exemples de stencil 5-points en 2D et 1D.

Applications stencils

Qu'est-ce qu'une application stencil ?

- Stencil : motif ou pochoir que l'on applique par répétition sur l'ensemble du domaine étudié

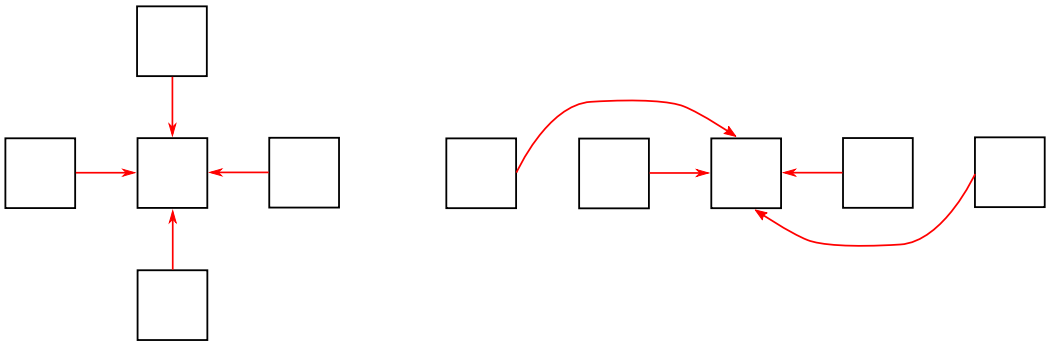


Figure: Exemples de stencil 5-points en 2D et 1D.

- Mise à jour d'une cellule en fonction du voisinage, schéma appliqué «simultanément» sur un ensemble de cellules

Applications stencils

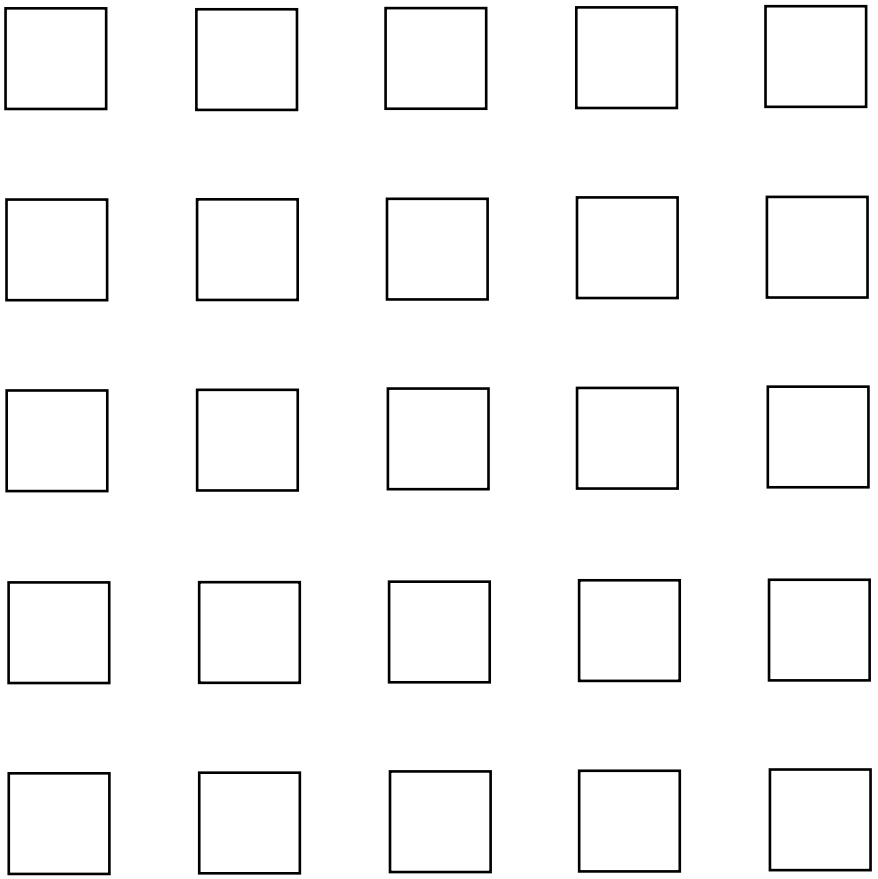


Figure: Exemple d'application de stencil.

Applications stencils

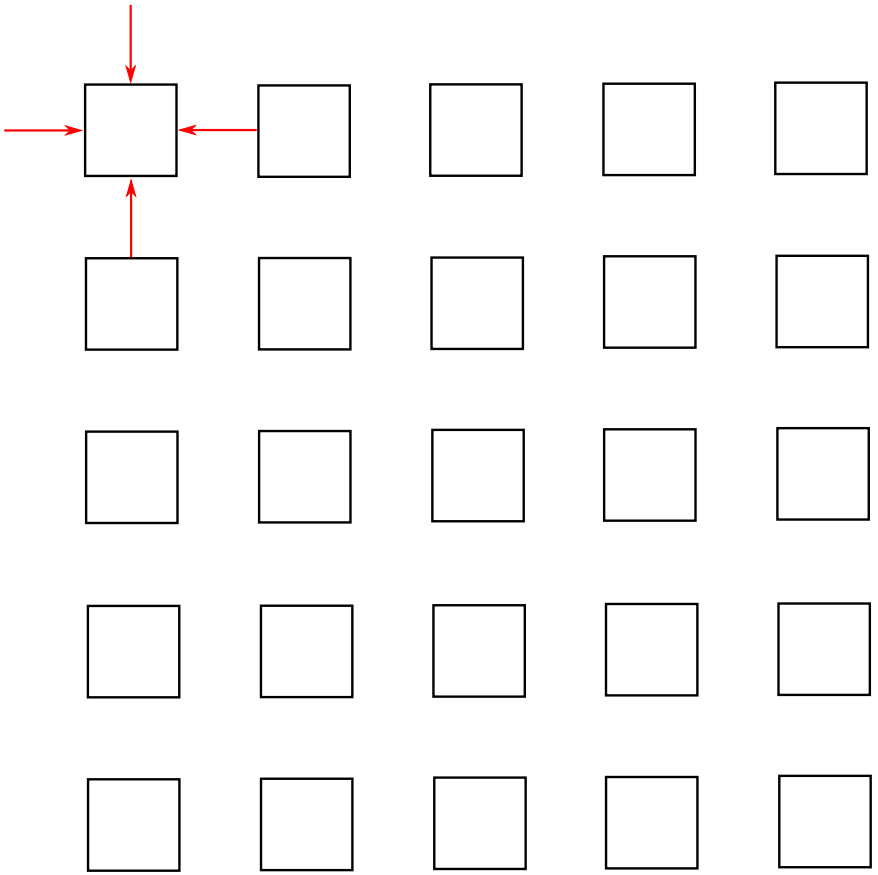


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé

Applications stencils

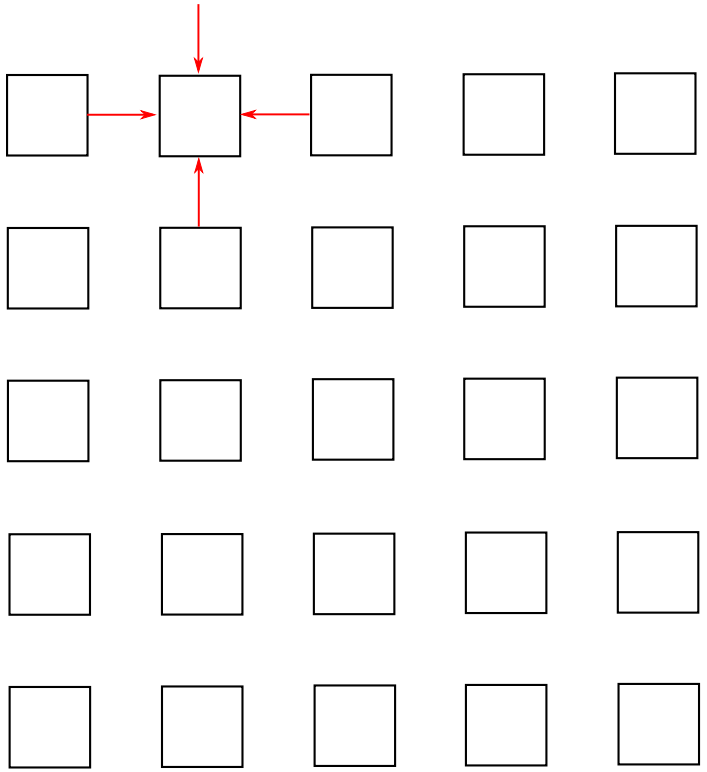


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé

Applications stencils

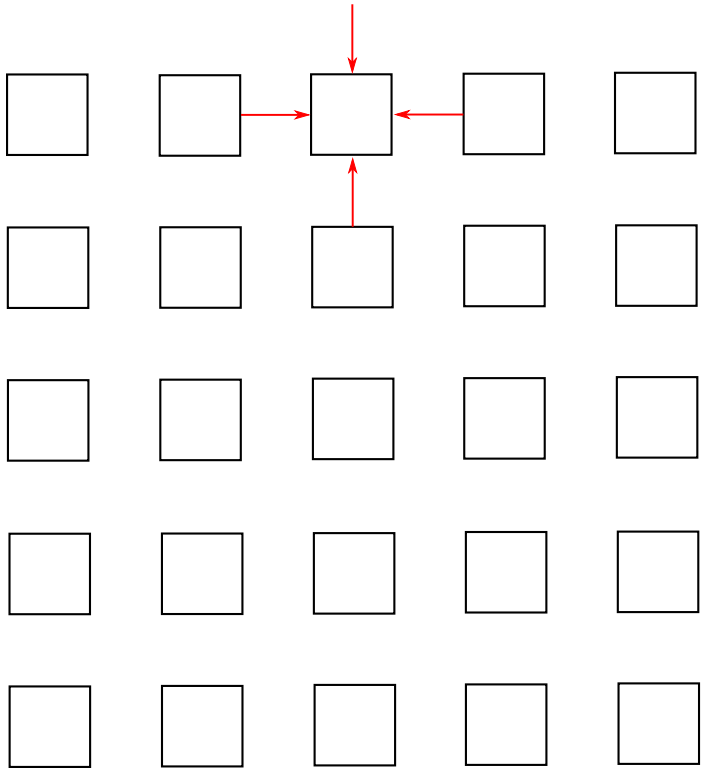


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé

Applications stencils

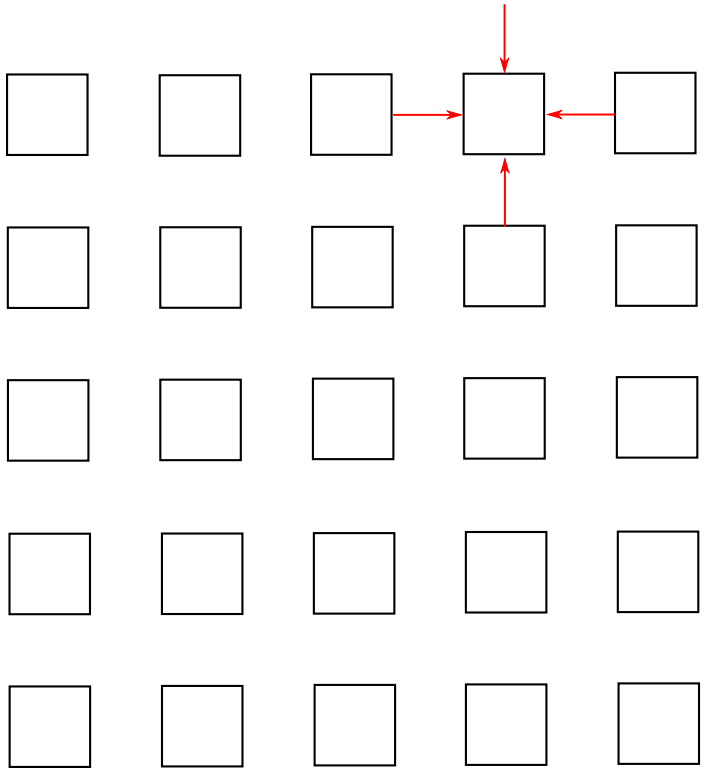


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé

Applications stencils

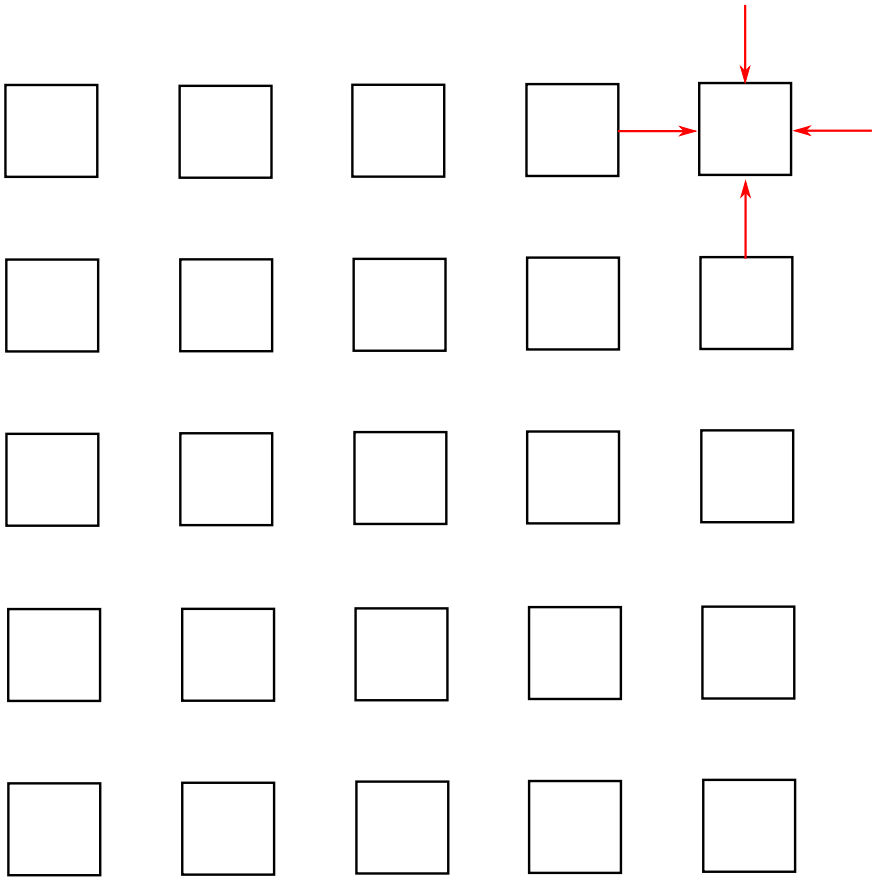


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé

Applications stencils

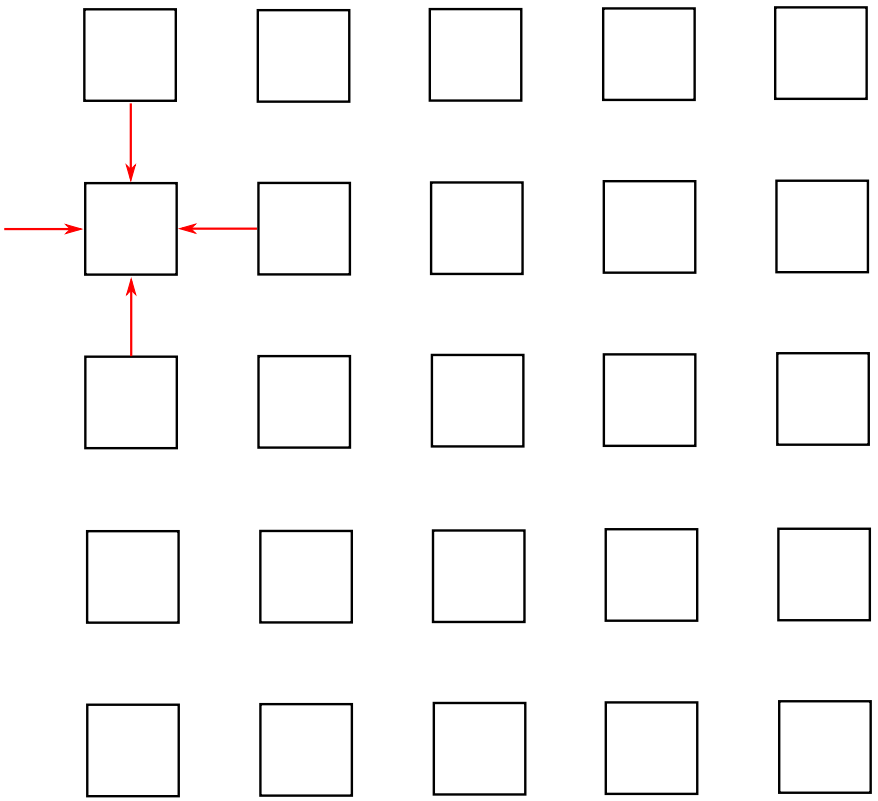


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé

Applications stencils

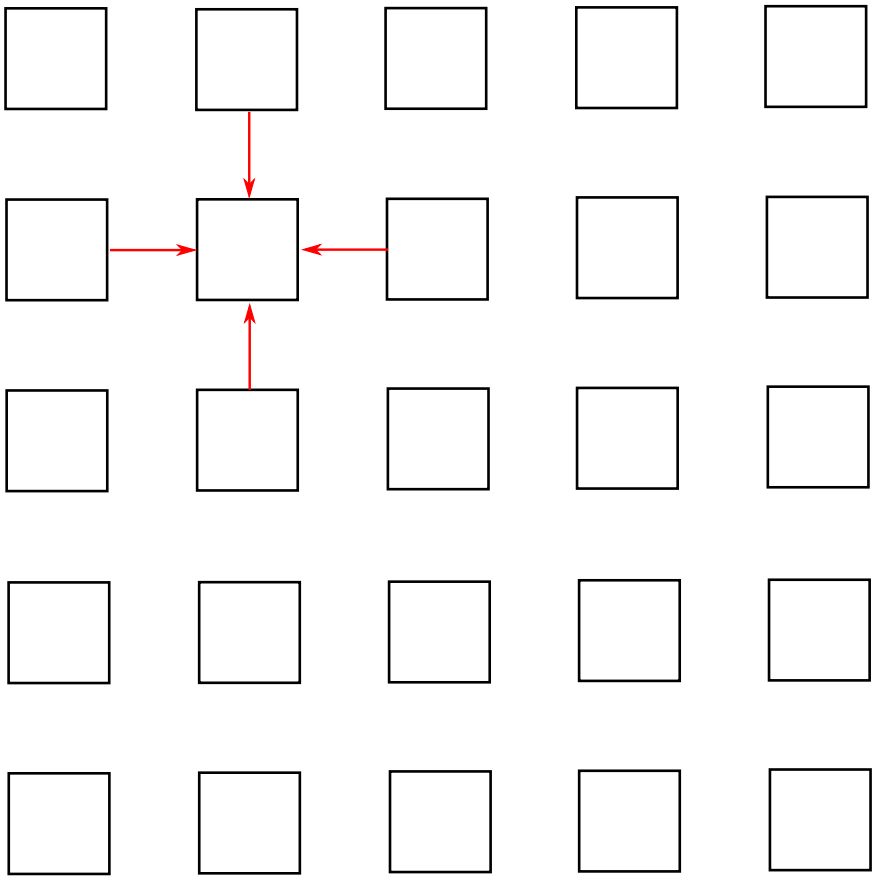


Figure: Exemple d'application de stencil.

- Ratio nombre d'accès mémoires par mise à jour très élevé
- Dimension temporelle construite avec des *itérations*

Objectifs généraux du stage

- Travail sur le recouvrement des transferts mémoires par du calcul

Objectifs généraux du stage

- Travail sur le recouvrement des transferts mémoires par du calcul
- Travail sur la localité (spatiale et temporelle) des données

Objectifs généraux du stage

- Travail sur le recouvrement des transferts mémoires par du calcul
- Travail sur la localité (spatiale et temporelle) des données
- Problématique du stage :
  Est-ce que les stratégies d'ordonnancement génériques du support d'exécution *StarPU* sont adaptées à des applications stencil, où la localité des données est cruciale ?

Environnement de test : exécution simulée

- reproductible
- total contrôle sur les paramètres :
  - durée d'une tâche
  - temps de transfert
  - architecture de la machine

Résumé de la contribution

- Outil de visualisation pour observer la localité des données

Résumé de la contribution

- Outil de visualisation pour observer la localité des données
- Méthode de référence : construction d'un ordre de soumission de tâches optimisé pour stencil

Résumé de la contribution

- Outil de visualisation pour observer la localité des données
- Méthode de référence : construction d'un ordre de soumission de tâches optimisé pour stencil
- Évaluation des ordonnanceurs de *StarPU* pour des applications stencils :

Résumé de la contribution

- Outil de visualisation pour observer la localité des données
- Méthode de référence : construction d'un ordre de soumission de tâches optimisé pour stencil
- Évaluation des ordonnanceurs de *StarPU* pour des applications stencils :
  - temps de calcul d'une tâche = 2 x temps de transfert

Résumé de la contribution

- Outil de visualisation pour observer la localité des données
- Méthode de référence : construction d'un ordre de soumission de tâches optimisé pour stencil
- Évaluation des ordonnanceurs de *StarPU* pour des applications stencils :
  - temps de calcul d'une tâche = 2 x temps de transfert
  - taille de problème > mémoire des cartes graphiques

Résumé de la contribution

- Outil de visualisation pour observer la localité des données
- Méthode de référence : construction d'un ordre de soumission de tâches optimisé pour stencil
- Évaluation des ordonnanceurs de *StarPU* pour des applications stencils :
  - temps de calcul d'une tâche = 2 x temps de transfert
  - taille de problème > mémoire des cartes graphiques
  - problème de déséquilibre de charge

Résumé de la contribution

- Outil de visualisation pour observer la localité des données
- Méthode de référence : construction d'un ordre de soumission de tâches optimisé pour stencil
- Évaluation des ordonnanceurs de *StarPU* pour des applications stencils :
  - temps de calcul d'une tâche = 2 x temps de transfert
  - taille de problème > mémoire des cartes graphiques
  - problème de déséquilibre de charge
- Assemblage d'un ordonnanceur qui exploite l'information de localité des données

Le cas à éviter : aucune localité des données

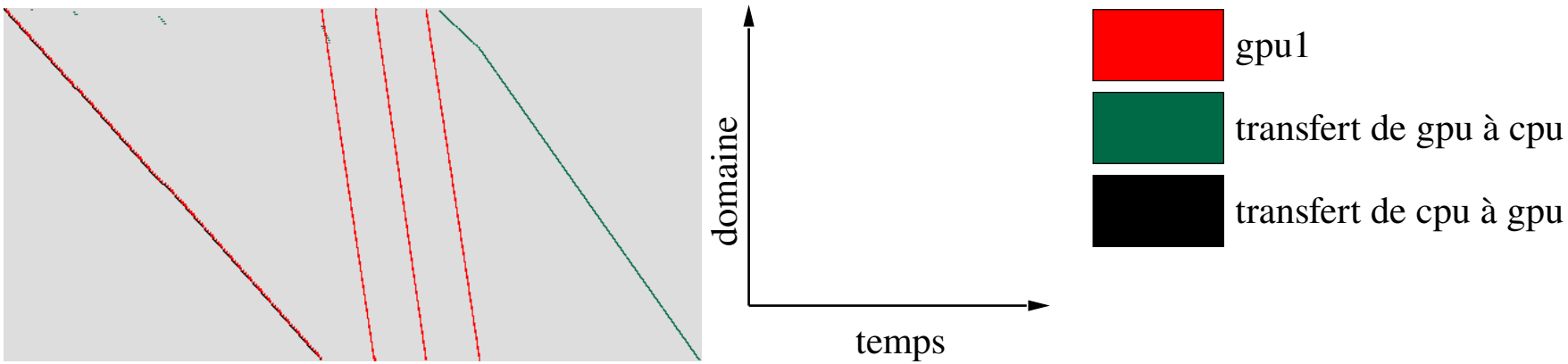


Figure: Diagramme en temps d'une exécution avec soumission de tâches naïve sur un GPU.

Le cas à éviter : aucune localité des données

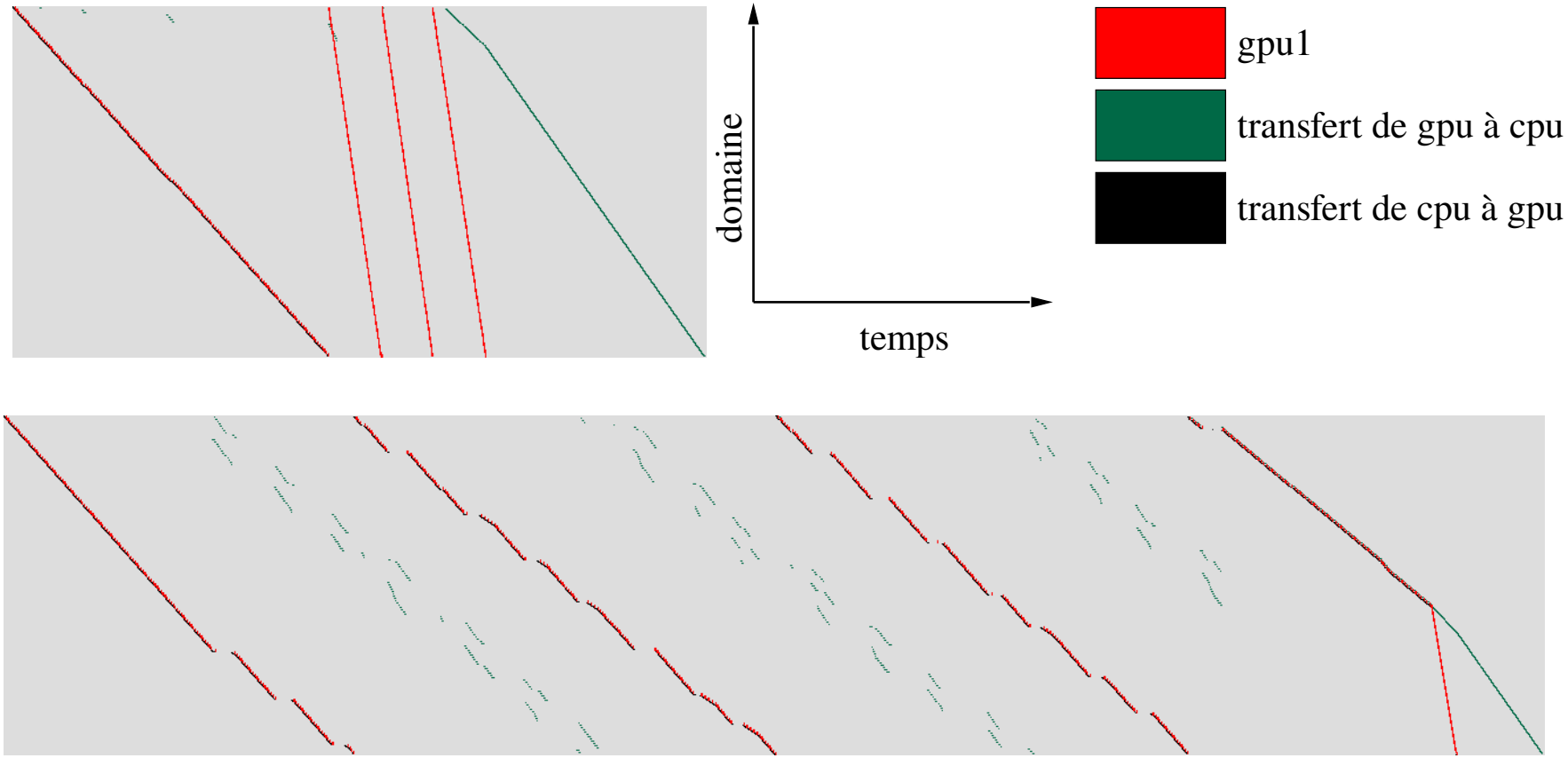


Figure: Diagramme en temps d'une exécution avec soumission de tâches naïve sur un GPU.

Algorithme cache oublieux (*oblivious*)

- Matteo FRIGO et Volker STRUMPEN : « Cache Oblivious Stencil Computations »

Algorithme cache oublieux (*oblivious*)

- Matteo FRIGO et Volker STRUMPEN : « Cache Oblivious Stencil Computations »

- Limite les chargements mémoires hors cache (*cache misses*)

Algorithme cache oublieux (*oblivious*)

- Matteo FRIGO et Volker STRUMPEN : « Cache Oblivious Stencil Computations »
- Limite les chargements mémoires hors cache (*cache misses*)
- Favorise la réutilisation des données

Algorithme cache oublieux (*oblivious*)

- Matteo FRIGO et Volker STRUMPEN : « Cache Oblivious Stencil Computations »
- Limite les chargements mémoires hors cache (*cache misses*)
- Favorise la réutilisation des données
- Découple l'optimisation du paramètre «taille du cache»

Soumission de tâches cache oublieux

Une soumission cache oublieuse de tâches au support d'exécution permet de :

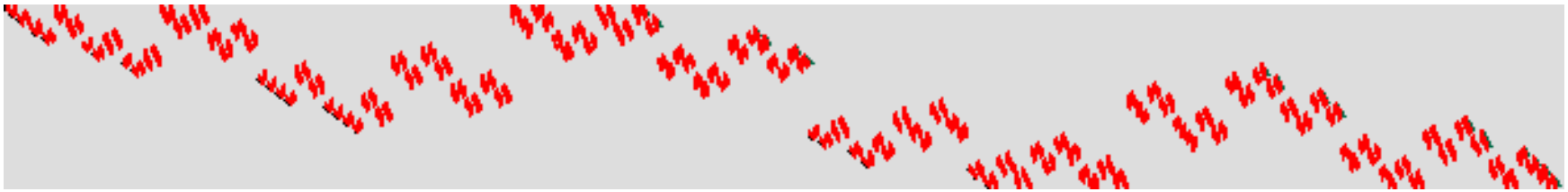
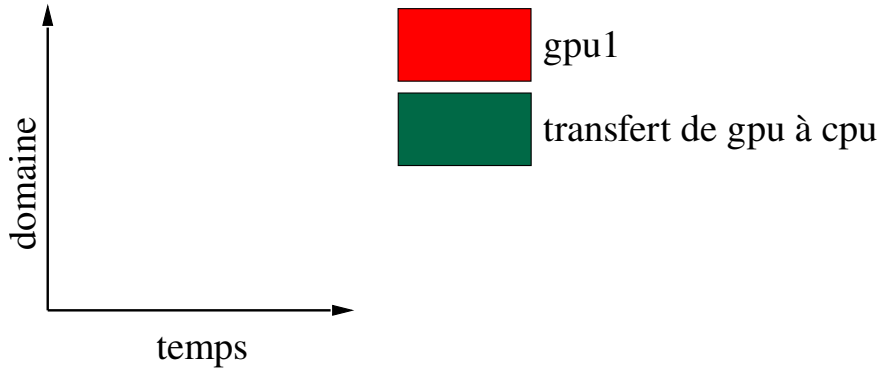


Figure: Diagramme en temps d'une exécution avec soumission de tâches cache oublieux sur un GPU.



Soumission de tâches cache oublieux

Une soumission cache oublieuse de tâches au support d'exécution permet de :

- Favoriser la localité des données

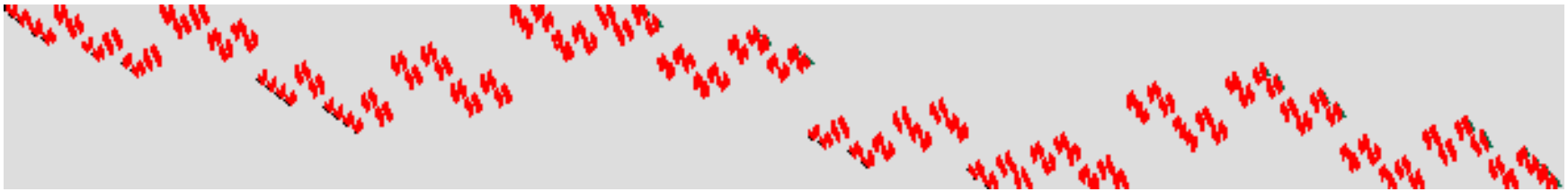
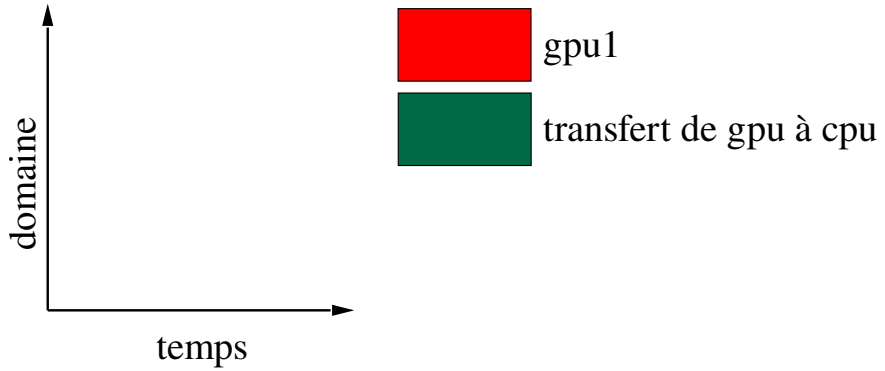


Figure: Diagramme en temps d'une exécution avec soumission de tâches cache oublieux sur un GPU.



Soumission de tâches cache oublieux

Une soumission cache oublieuse de tâches au support d'exécution permet de :

- Favoriser la localité des données
- Limiter les transferts mémoires entre cartes graphiques et mémoire principale

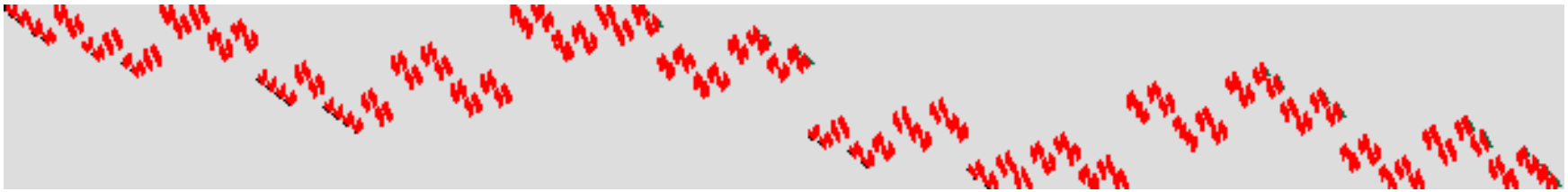
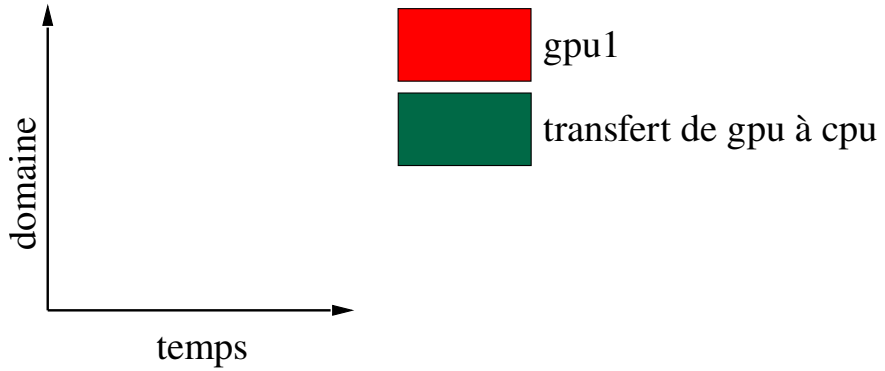


Figure: Diagramme en temps d'une exécution avec soumission de tâches cache oublieux sur un GPU.



Version parallèle - respect des dépendances

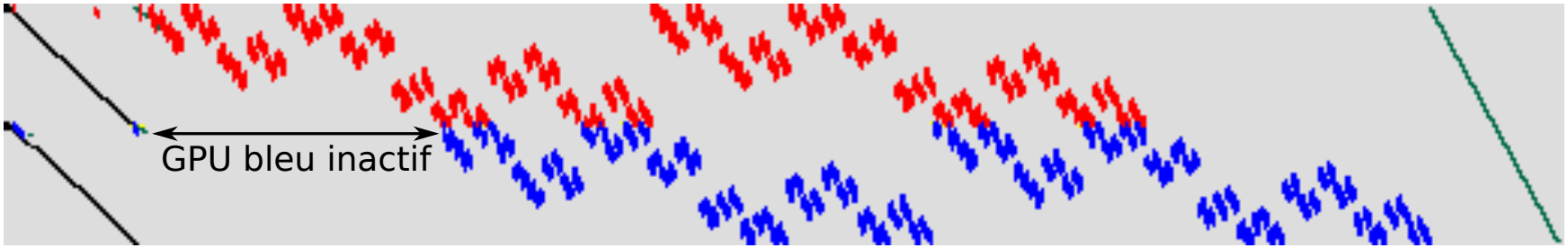
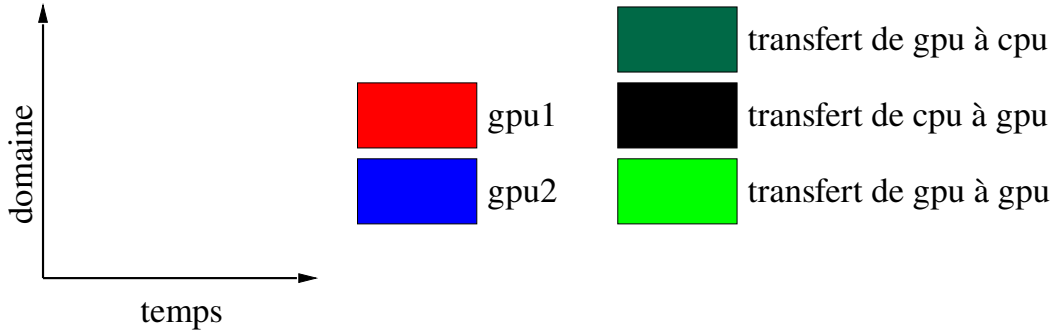


Figure: Diagramme en temps d'une exécution avec soumission parallèle de tâches cache oublieux sur deux GPU.



Version parallèle - respect des dépendances

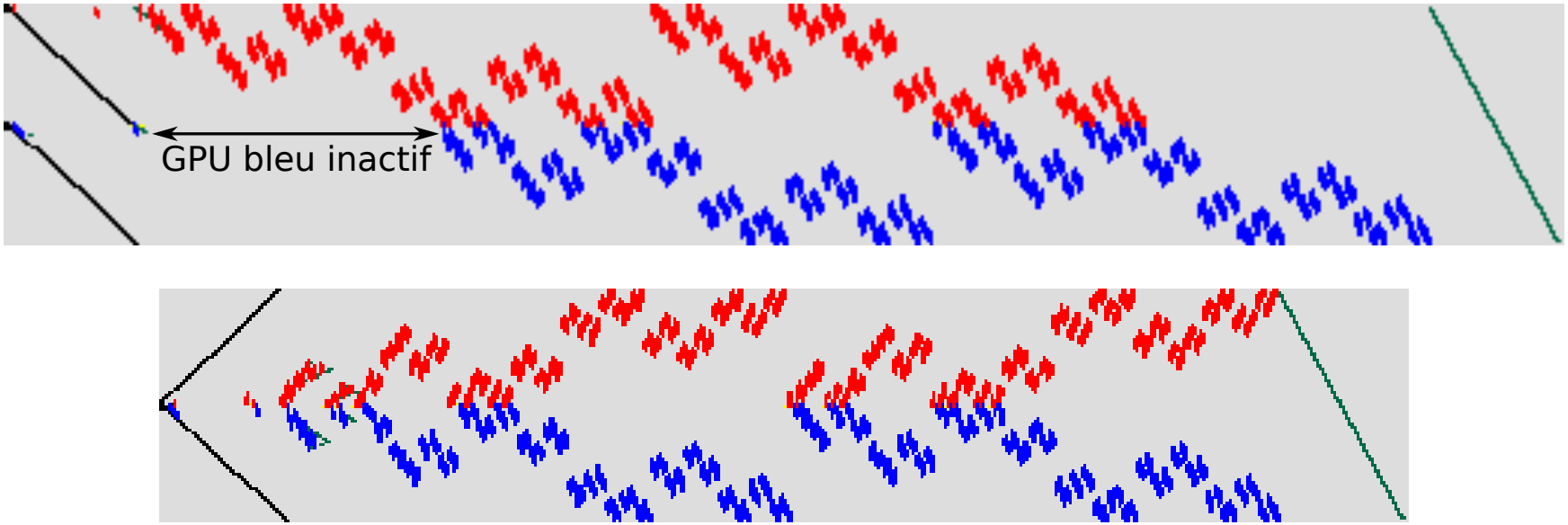
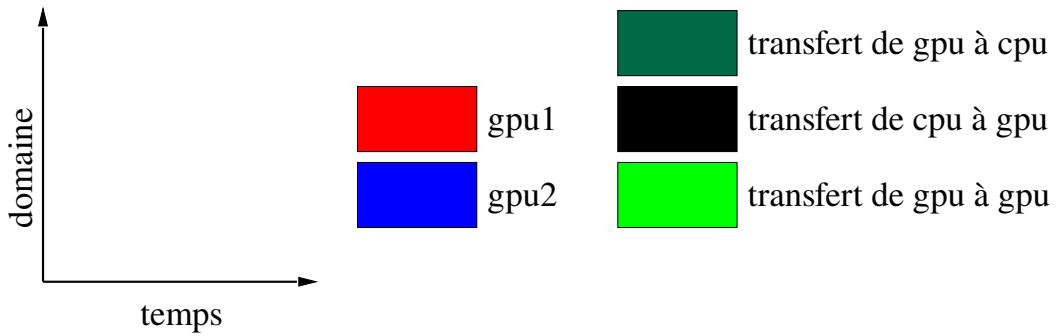
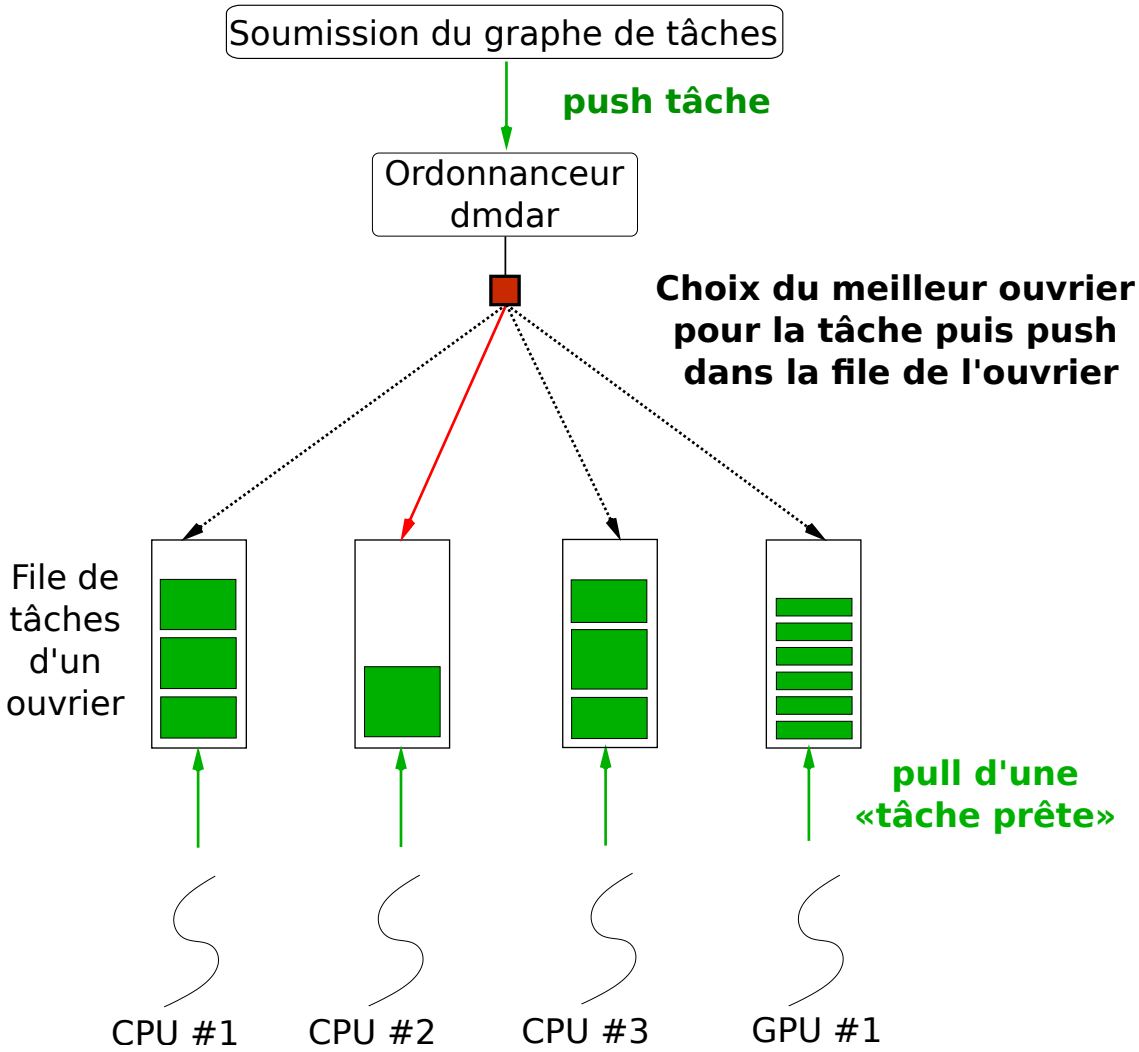


Figure: Diagramme en temps d'une exécution avec soumission parallèle de tâches cache oublieux sur deux GPU.

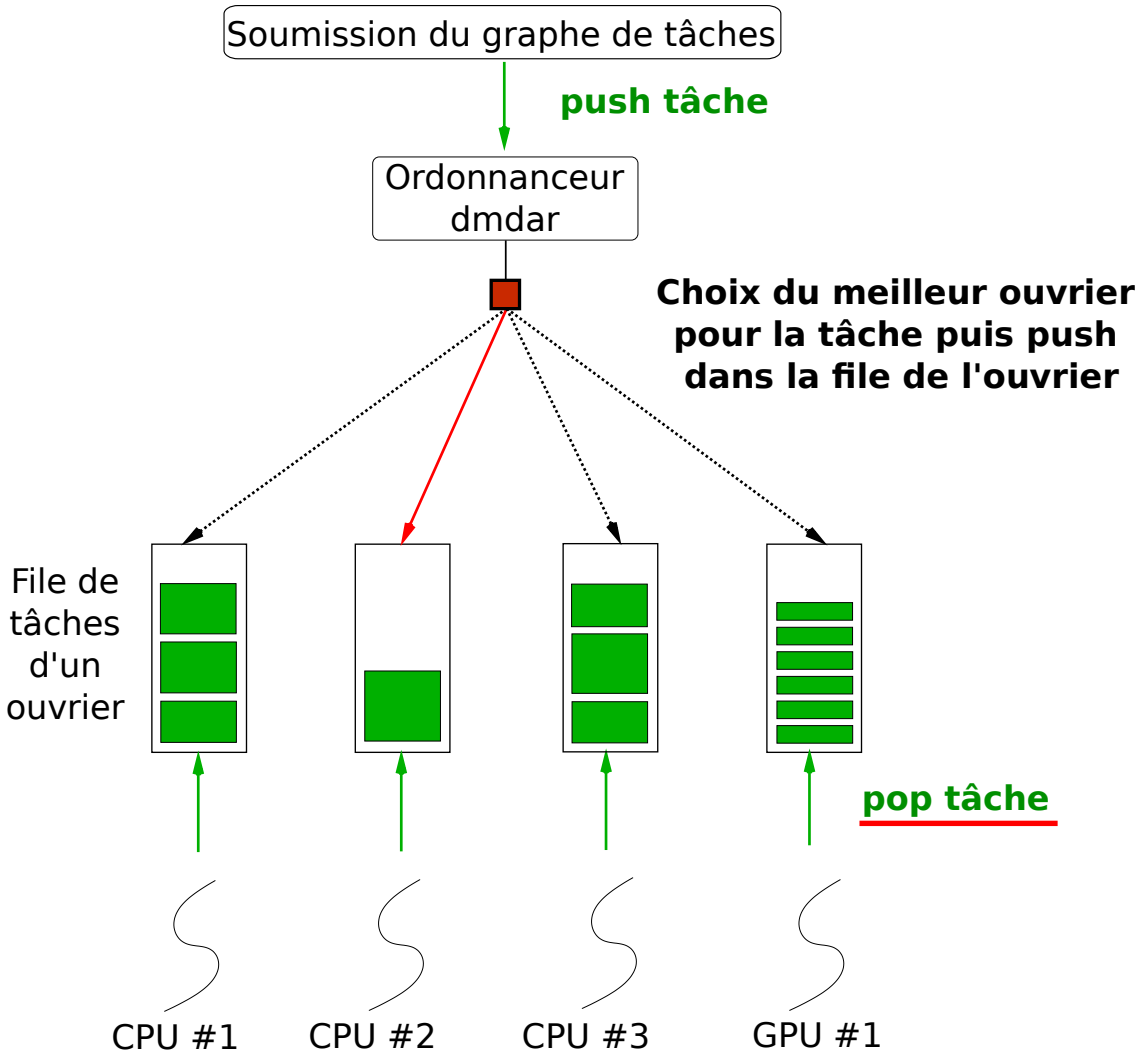


Ordonnancement d'une tâche pour l'ordonnanceur `dmdar`



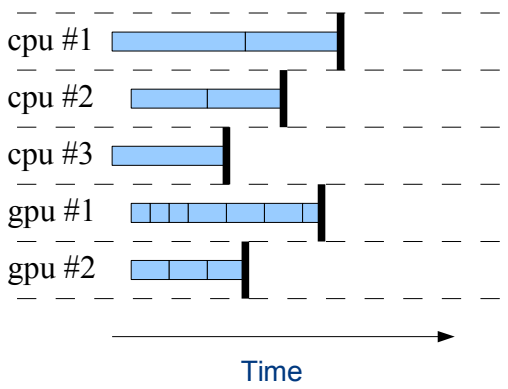
- Modèle de performance (temps de calcul des tâches)
- Temps de transfert
- Tri des tâches par données prêtes

Ordonnancement d'une tâche pour l'ordonnanceur `dmda`

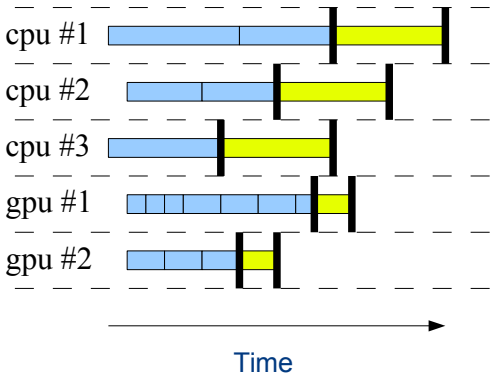


- Modèle de performance (temps de calcul des tâches)
- Temps de transfert

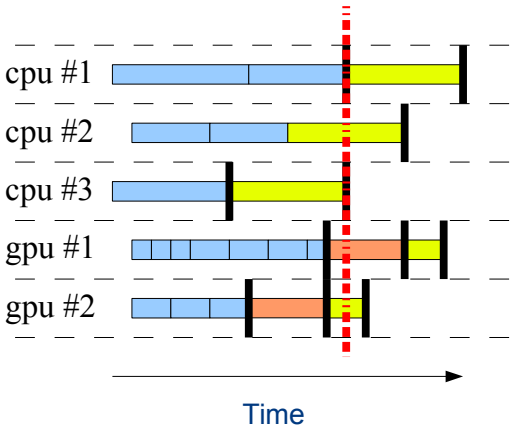
Choix du meilleur ouvrier pour une tâche



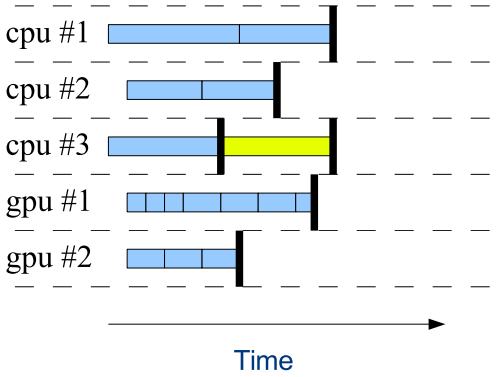
(a) État initial des files de tâches.



(b) Temps de calcul.



(c) Temps de transfert.



(d) Choix de l'ouvrier idéal.

Figure: Recherche d'une date de terminaison minimale.

Comparaison avec la méthode de référence

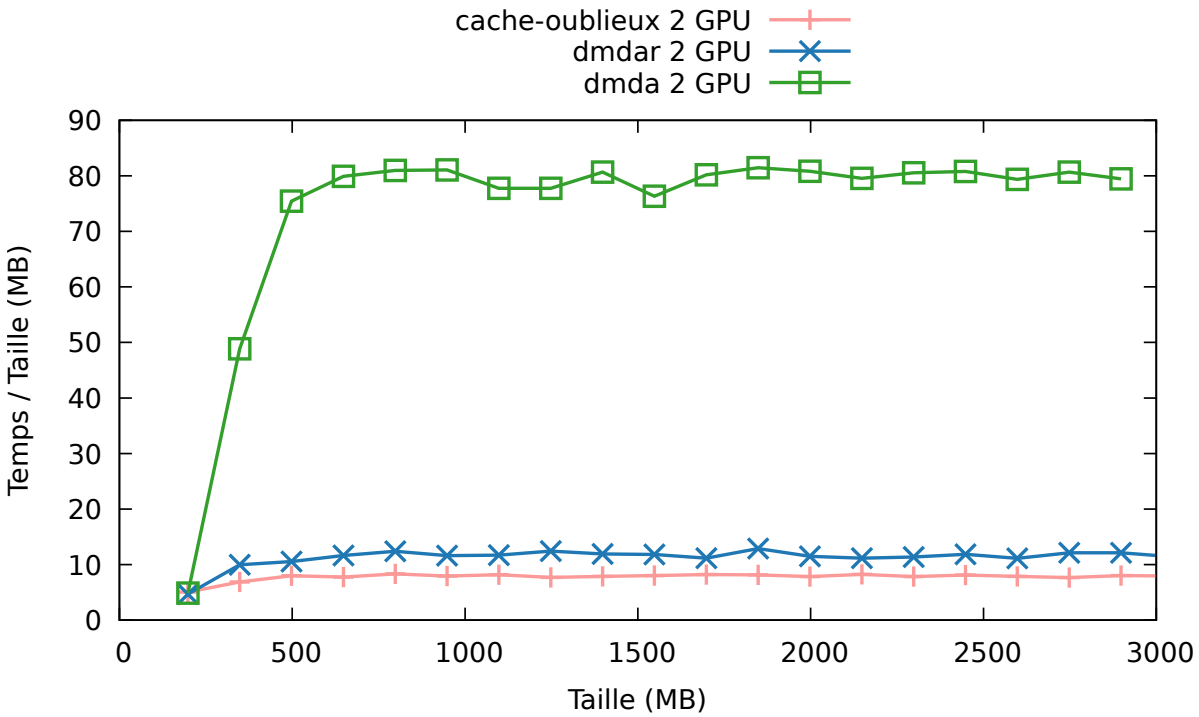


Figure: Évolution du temps d'exécution en fonction de la taille du problème - 2 GPU, limite mémoire fixée à 200MB par GPU.

Comparaison avec la méthode de référence

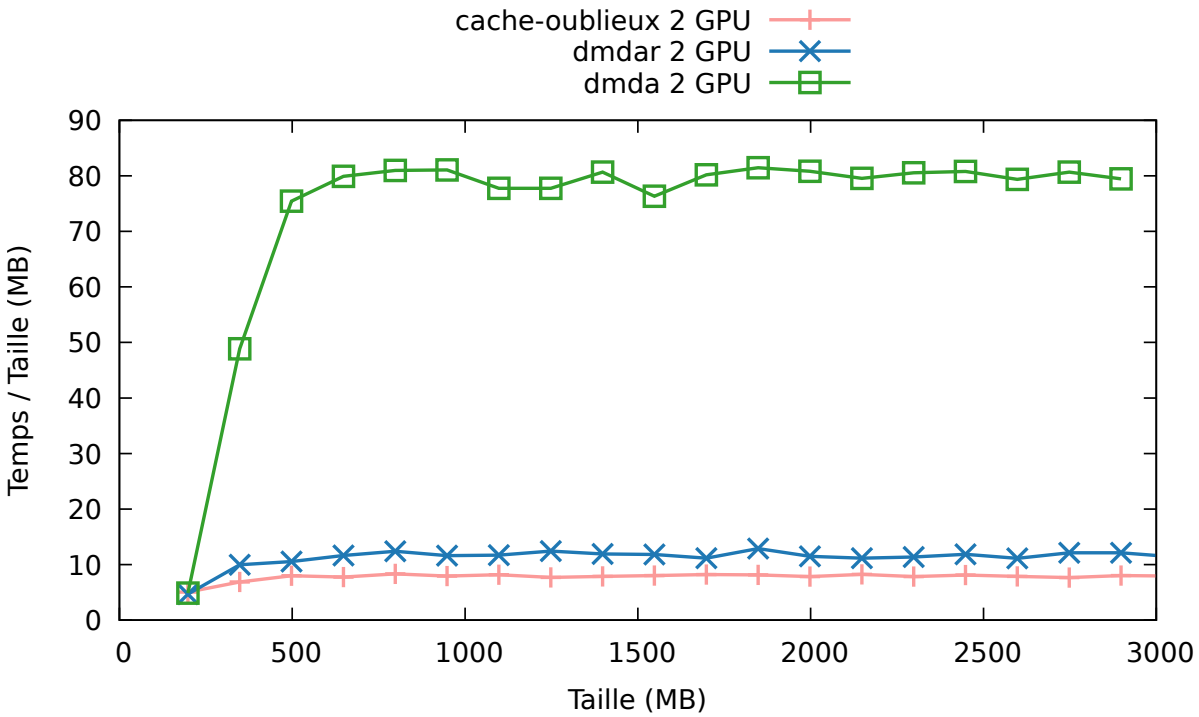


Figure: Évolution du temps d'exécution en fonction de la taille du problème - 2 GPU, limite mémoire fixée à 200MB par GPU.

- Souligne l'importance du recouvrement des transferts mémoires

Comparaison avec la méthode de référence

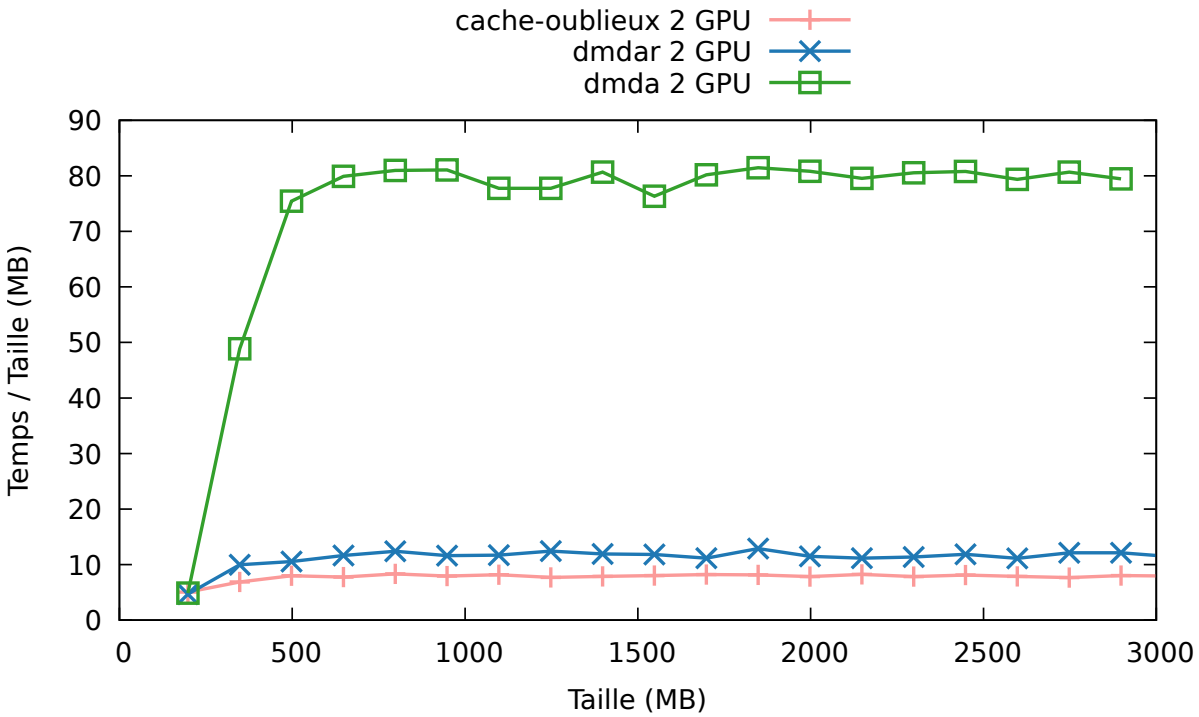


Figure: Évolution du temps d'exécution en fonction de la taille du problème - 2 GPU, limite mémoire fixée à 200MB par GPU.

- Souligne l'importance du recouvrement des transferts mémoires
- `dmdar` ne casse pas les situations favorables

Comparaison avec la méthode de référence

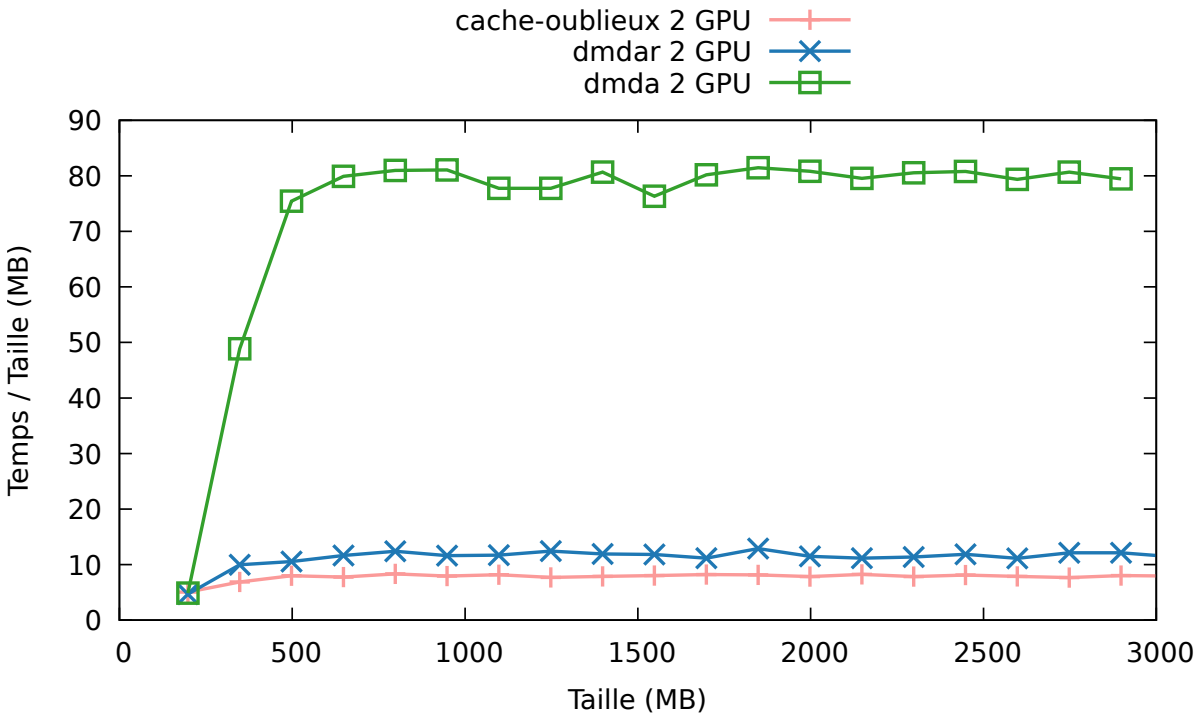
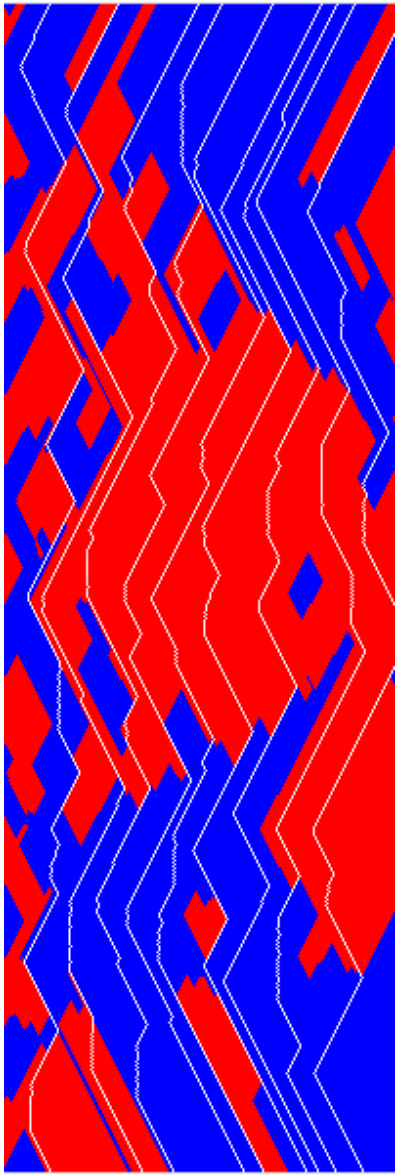


Figure: Évolution du temps d'exécution en fonction de la taille du problème - 2 GPU, limite mémoire fixée à 200MB par GPU.

- Souligne l'importance du recouvrement des transferts mémoires
- `dmdar` ne casse pas les situations favorables
- Performances encore trop éloigné de l'idéal (1.5x plus lent)



Objectif : limiter la quantité de frontières

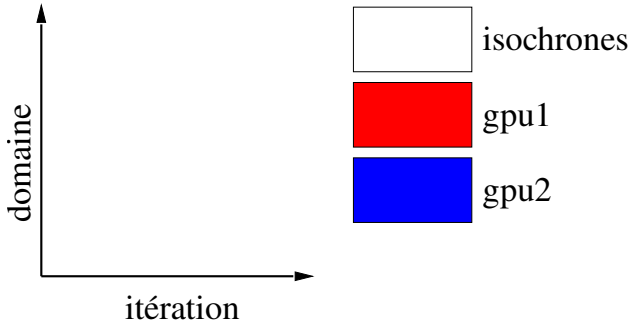


Figure: Diagramme en itérations d'une exécution de `dmdar`. 2 GPU, taille du problème 1800MB, limite mémoire fixée à 200MB par GPU.

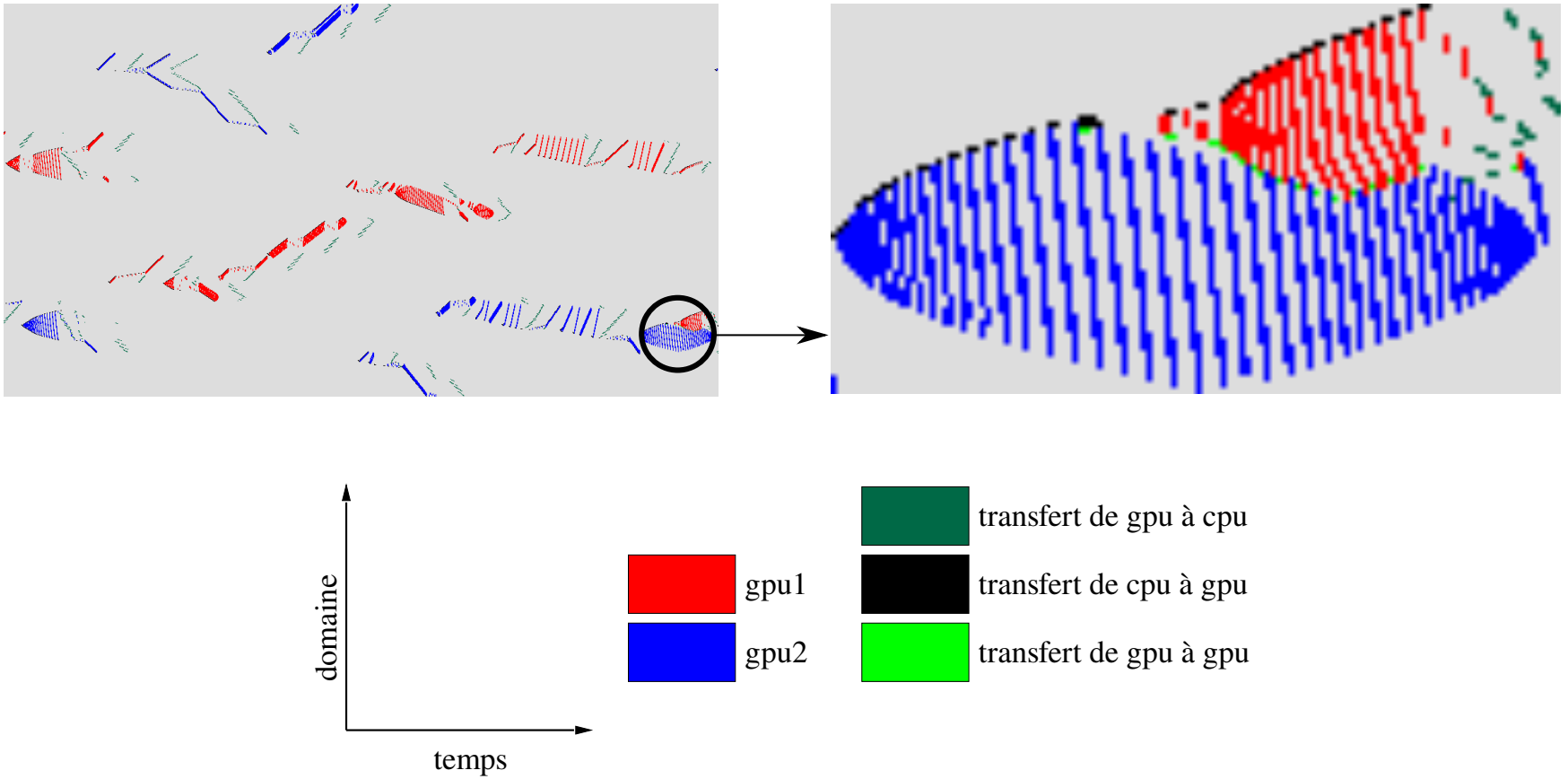


Figure: Diagramme en temps - Échantillon d'exécution de `dmdar`. 2 GPU, taille du problème 1800MB, limite mémoire fixée à 200MB par GPU.

Résumé des points importants

- Stencil :
  - Ratio accès mémoire par mises à jour élevé
  - Observer la localité des données : outil adapté
- Évaluer les ordonnanceurs :
  - Méthode de référence : algorithme cache oublieux pour stencil
  - Est-ce que les ordonnanceurs (génériques) de *StarPU* sont adaptés à des stencils ?

Conclusion

Bilan :

- Le support d'exécution *StarPU*, performances satisfaisantes :
  - encore trop éloignées de l'idéal
  - générique et portable

Améliorations :

- Ordonnanceur : améliorer l'amorce
- Visualisation pour stencils à dimension arbitraire (2D, 3D, etc.)
- Test d'un vrai stencil (e.g. application de simulation nucléaire)

Perspectives

- *Out of core* : plus de place en mémoire principale
  - Rapatriement en mémoire du disque dur

- Programmation distribuée en réseau (*Message Passing Interface*)