Intégration d'un ordonnanceur pour architectures hétérogènes dans PaStiX-StarPU

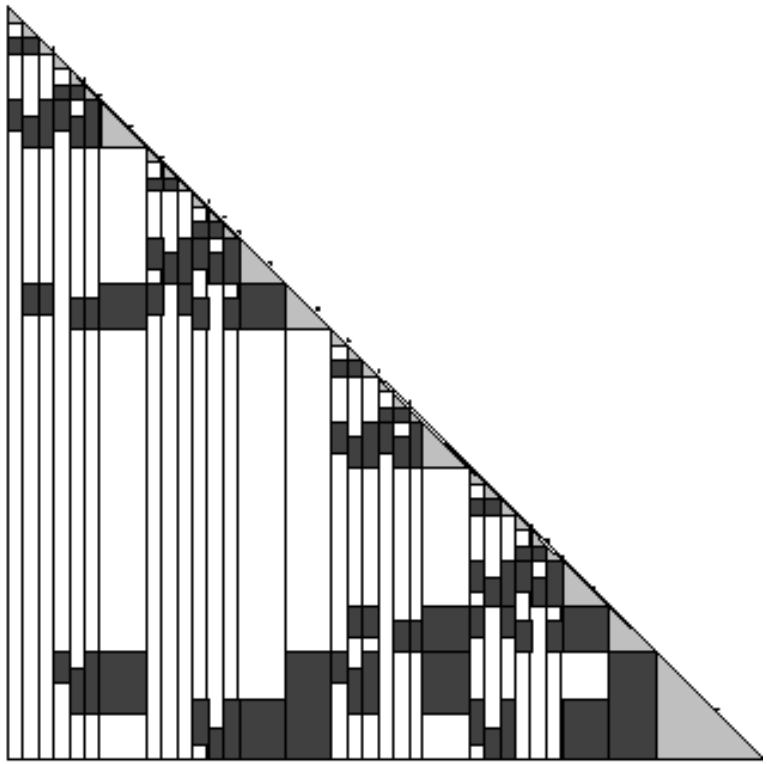
Equipe HiePACS

Moënne-Loccoz Tom

Sommaire

PaStiX

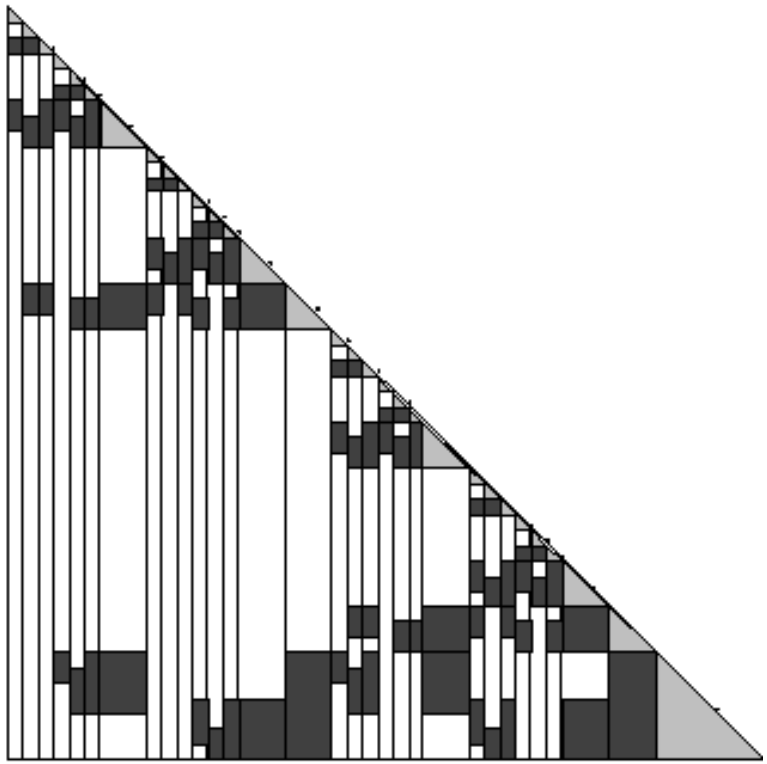
Solveur linéaire creux.



PaStiX

Solveur linéaire creux.

- Parallèle
- Distribué
- Hétérogène

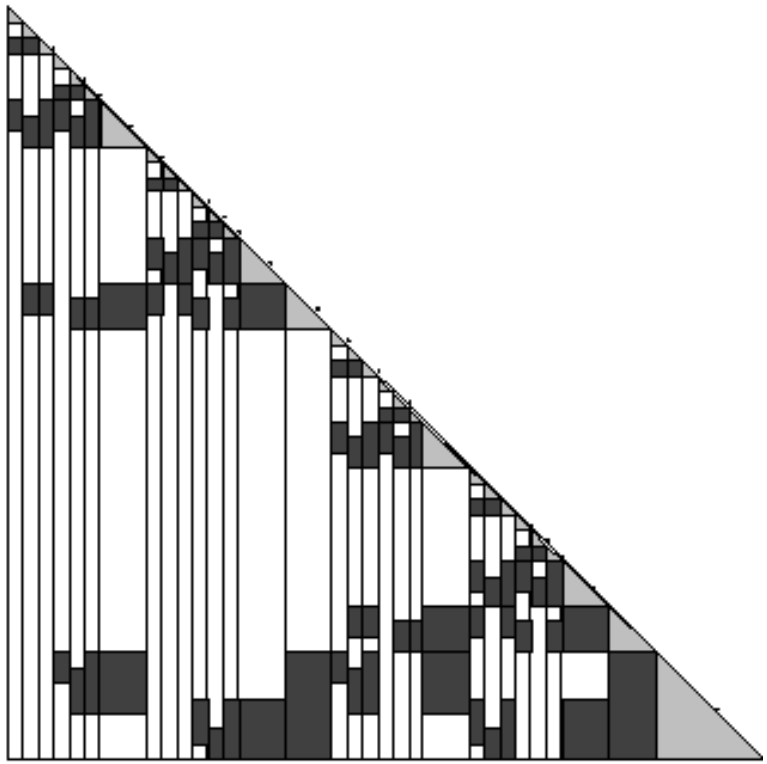


PaStiX

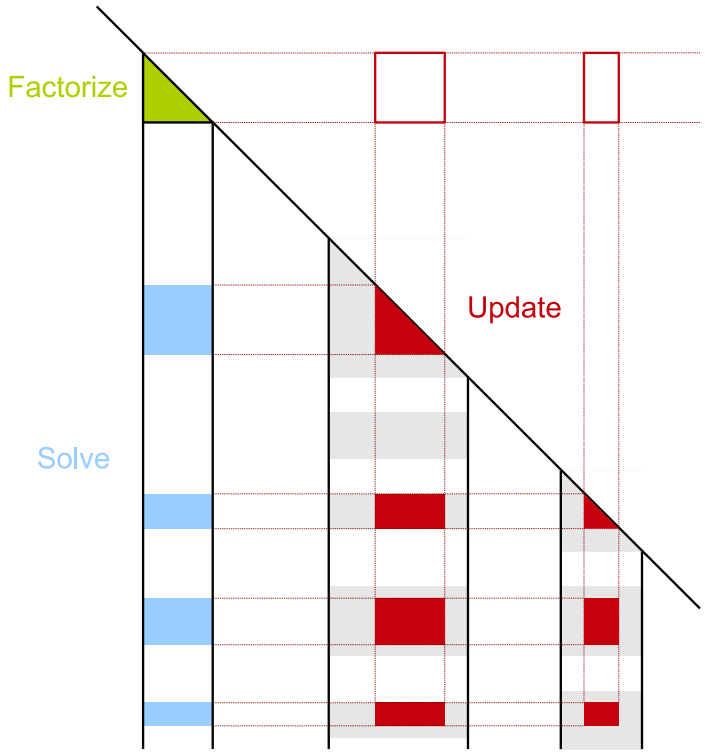
Solveur linéaire creux.

- Parallèle
- Distribué
- Hétérogène

Fonctionne par factorisation.

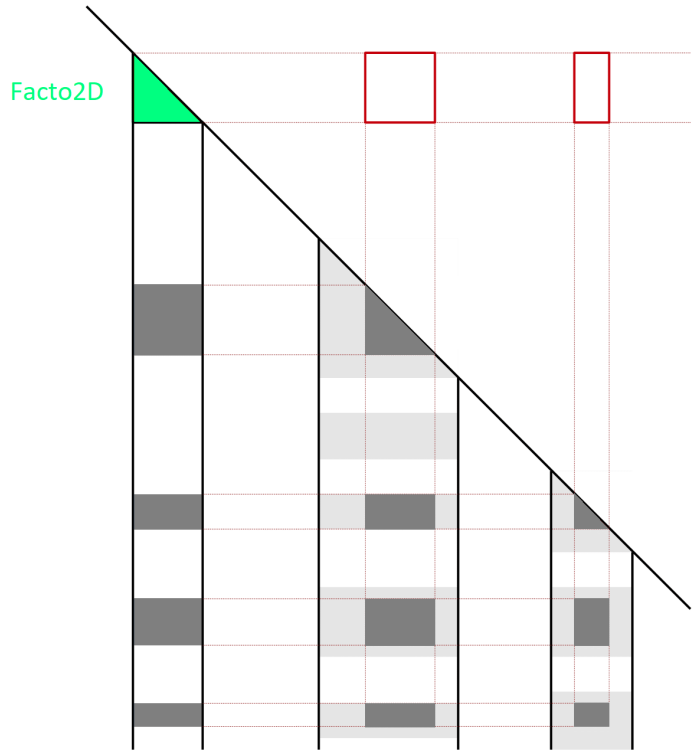


Factorisation numérique



Kernels :

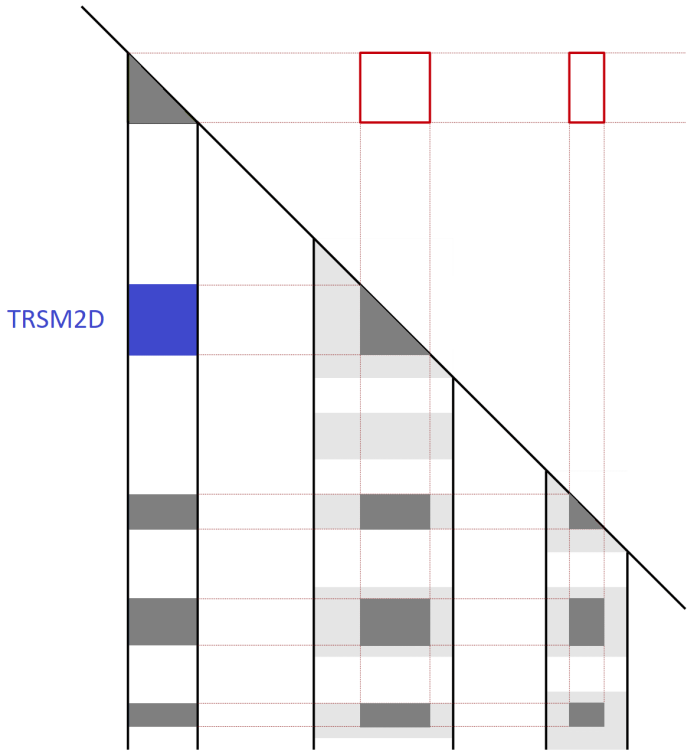
Factorisation numérique



Kernels :

- Au niveau des blocs :
  - Facto2D

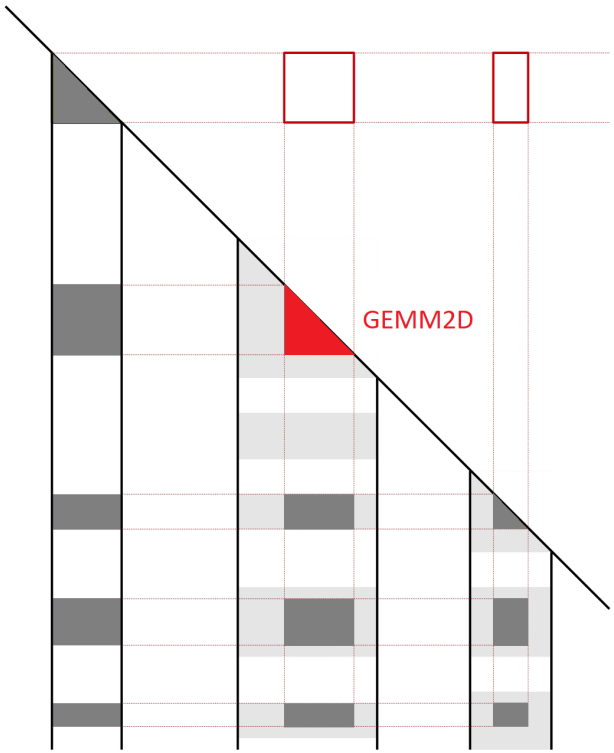
Factorisation numérique



Kernels :

- Au niveau des blocs :
  - Facto2D
  - TRSM2D

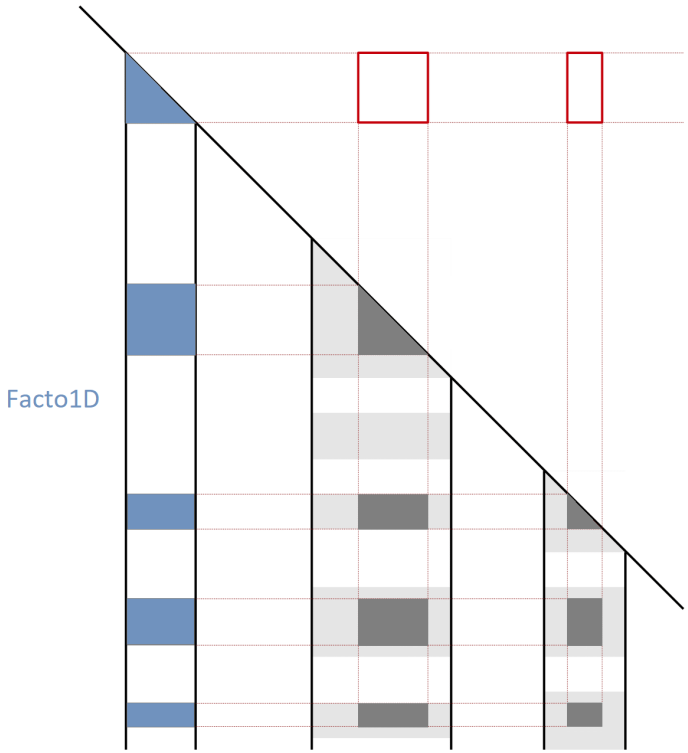
Factorisation numérique



Kernels :

- Au niveau des blocs :
  - Facto2D
  - TRSM2D
  - GEMM2D

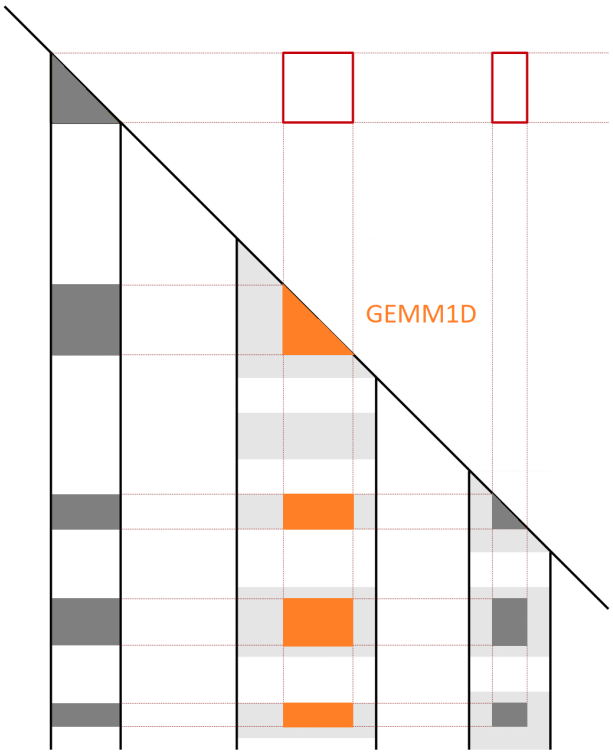
Factorisation numérique



Kernels :

- Au niveau des blocs :
  - Facto2D
  - TRSM2D
  - GEMM2D
- Au niveau des colonnes entières:
  - Facto1D

Factorisation numérique



Kernels :

- Au niveau des blocs :
  - Facto2D
  - TRSM2D
  - GEMM2D
- Au niveau des colonnes entières:
  - Facto1D
  - GEMM1D

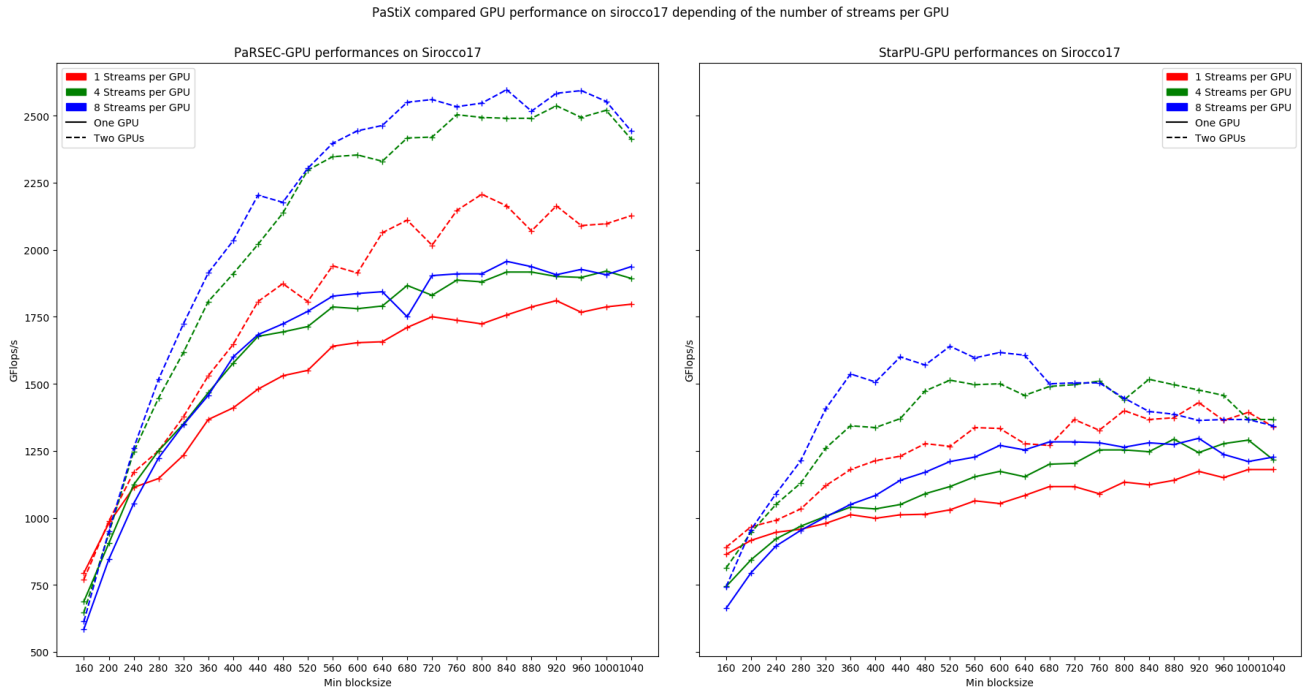
Supports d'exécution

StarPU et ParSEC

- Abstraient les détails de l'architecture.
- Gèrent les dépendances des tâches.
- Facilitent l'hétérogénéité.
- Intègrent un ordonnaceur (dmdas pour StarPU).

Problème

PaStiX-StarPU exploitent moins bien les cartes graphiques que PaStiX-PaRSEC:



Heteroprio

Algorithme HETEROPRIO

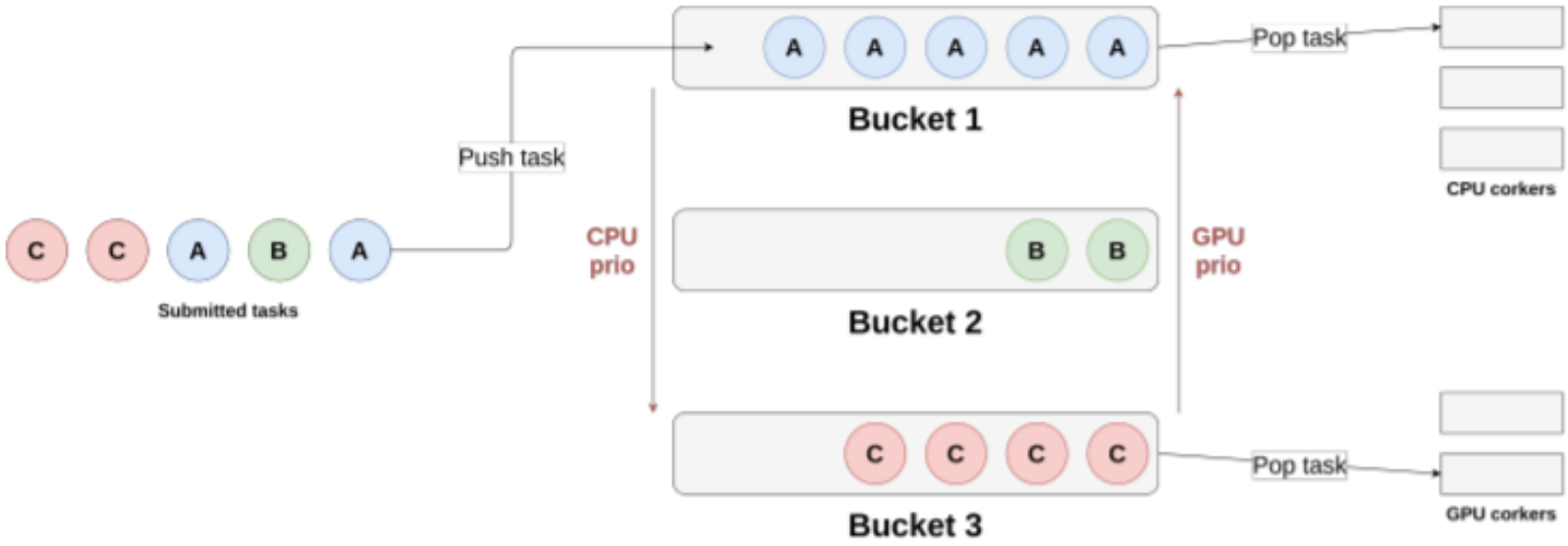
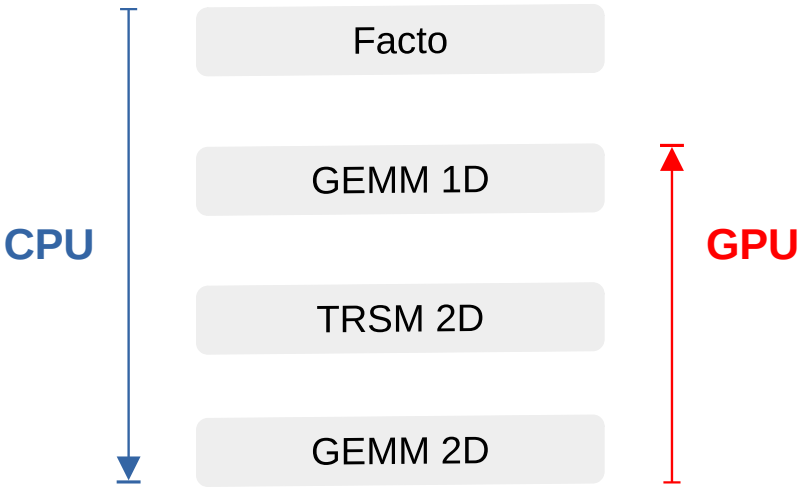


Figure: Schema de principe de HETEROPRIO

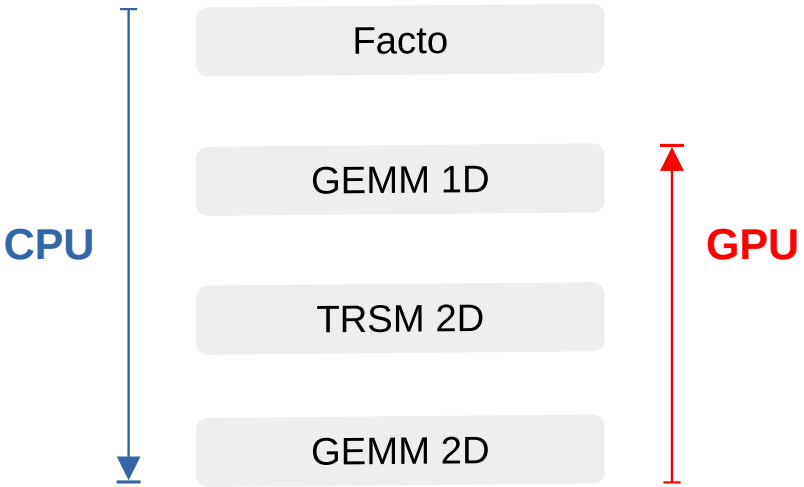
Mappings proposés

1 bucket par type de kernel :

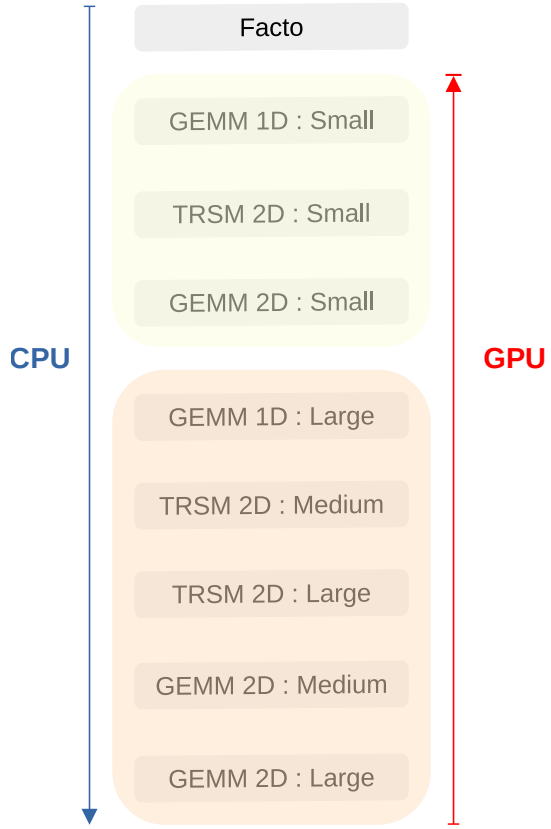


Mappings proposés

1 bucket par type de kernel :



2/3 bucket par type de kernel :



Facteurs d'accélération proposés

- Facteur théorique :

$$\frac{\text{Vitesse théorique de calcul GPU}}{\text{Vitesse théorique de calcul CPU}}$$

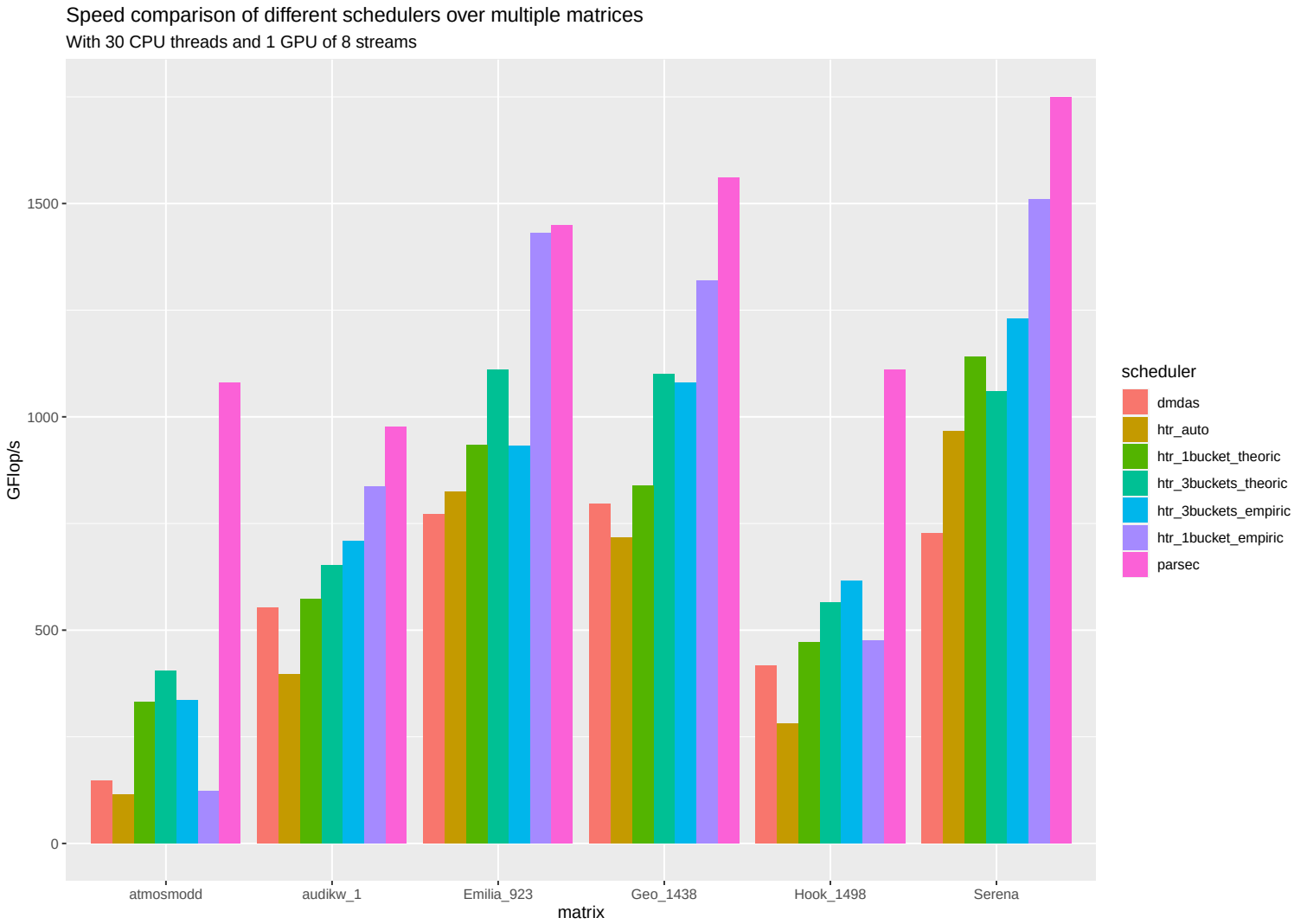
Facteurs d'accélération proposés

- Facteur théorique :

$$\frac{\text{Vitesse théorique de calcul GPU}}{\text{Vitesse théorique de calcul CPU}}$$

- Facteur empirique : Extrait d'un logging de chaque tâche et du calcul du ratio associé.
  Demande une étude statistique plus précise pour le mapping 2/3 buckets.

Résultats


Speed comparison of different schedulers over multiple matrices
With 30 CPU threads and 1 GPU of 8 streams
GFlop/s
1500
1000
500
0
atmosmodd
audikw_1
Emilia_923
Geo_1438
Hook_1498
Serena
matrix
scheduler
dmdas
htr_auto
htr_1bucket_theoric
htr_3buckets_theoric
htr_3buckets_empiric
htr_1bucket_empiric
parsec

Conclusion

- Un gain de performance important par rapport à l'ordonnanceur originale (dmdas) !

Conclusion

- Un gain de performance important par rapport à l'ordonnanceur originale (dmdas) !
- Toujours en dessous de PaRSEC...

Conclusion

- Un gain de performance important par rapport à l'ordonnanceur originale (dmdas) !
- Toujours en dessous de PaRSEC...
- Des pistes à creuser :
  - Analyse des traces pour comprendre ce qui pénalise sur les petites matrices.
  - Mapping visant plus le chemin critique.
  - Rendre dynamique l'extraction du ratio de performance pour les solutions théoriques permettrait d'avoir des performances portables.

Merci pour votre attention.
Avez-vous des questions ?