Portage efficace d'un code de calcul en mécanique des fluides sur GPUs sans coder une ligne de CUDA (ou presque)

Annaïg Pedrono, Aurélie Louis-Napoléon
Maxime Pigou, Pierre Elyakime (IMFT)

Mickaël Duval, Nicolas Renon (CALMIP)

JCAD 2019, mercredi 9 octobre 2019, Toulouse

INSTITUT DE MECANIQUE DES FLUIDES DE TOULOUSE

calmip

GENCI

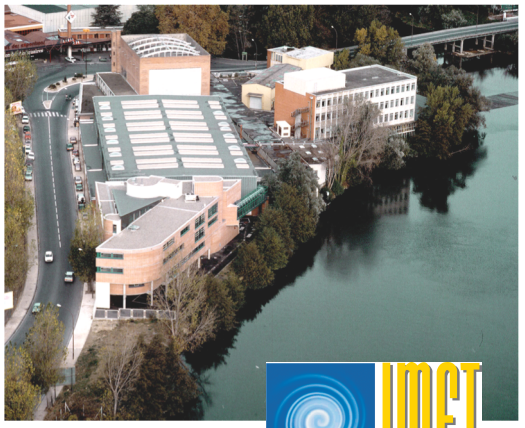
JÜLICH FORSCHUNGSZENTRUM

Plan de l'exposé

- Présentation du contexte
- Hackathon GENCI
- Hackathon GPU Oak Ridge
- Tests MPI Cuda-Aware
- Conclusion et perspectives

Institut de Mécanique des Fluides de Toulouse

Unité Mixte de Recherche
- 200 personnes
- 70 chercheurs et enseignants-chercheurs
- 35 personnels d'appui à la recherche
- une centaine de doctorants et post-docs.



Groupes de recherche

- **Aérodynamique, Sillages, Interactions** — Resp. : P. Brancher
- **Milieux Poreux et Biologiques** — Resp. : P. Swider
- **Milieux Réactifs** — Resp. : L. Selle
- **Fluides & Particules** — Resp. : P. Fede
- **Pôle Eco-Hydraulique** — Dir. : P. Sagnes
- **Interfaces** — Resp. : S. Tanguy
- **Hydrologie, Eco-Hydraulique et Ondes** — Resp. : H. Roux

Soutien aux activités numériques

- **Pôle Support / Proximité DSIN** — Resp. : Y. Exposito; C. Carlès, A. Campini
- **Pôle Infrastructures DSIN** — Resp. : C. Nicolas, A. Stoukov
- **Codes et Simulations Numériques Pôle Calculs & Données DSIN** — Resp. : H. Neau; P. Elyakime, A. Pedrono, M. Pigou, A. Stoukov

CALMIP

Mésocentre de calcul

Unité Mixte de Service du CNRS depuis 2014.

Renouvellement de son calculateur tous les 4 ans environ.

Depuis septembre 2018, le supercalculateur Olympe (1,365Pflops/s Peak) est en production.

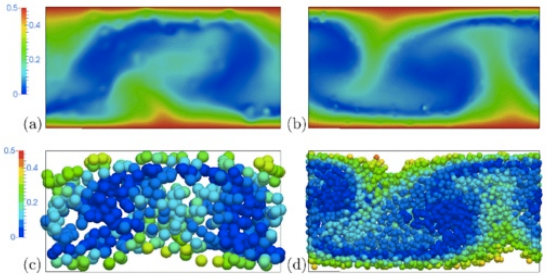
Une équipe de 7 ingénieurs pour l'exploitation du calculateur et le support technique et scientifique aux utilisateurs

Supercalculateur Olympe

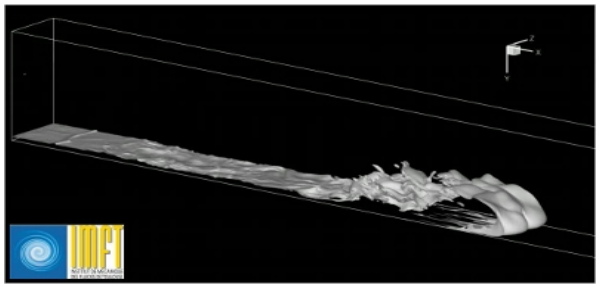
JADIM

Code de calcul volumes finis développé à l'IMFT et au LGC depuis 30 ans

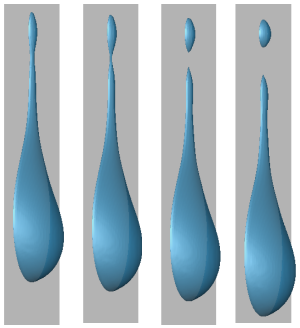
- Ecoulements instationnaires fluides incompressibles
- Maillages structurés cartésiens ou curvilignes
- Nombreux modules physiques dont la simulation des écoulements multiphasiques par une approche de type Volume Of Fluid ou Level-Set,
- Fortran 90/95 parallélisé avec une méthode de décomposition de domaines (MPI)
- Méthode de projection pour la pression : résolution du système linéaire creux (MUMPS et PETSc) => entre 75 et 95% du temps de calcul selon la physique et la taille du problème



Inertial finite-size particles in turbulent Couette flow (Force Coupling Method). G. Wang.



Gravity current. Y. Hallez



Sliding drop. M. Maglio

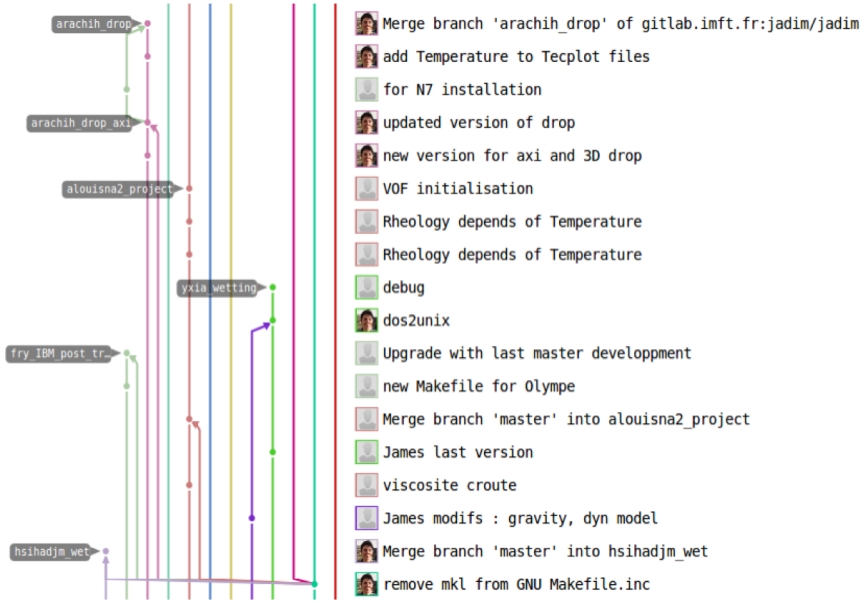
Bubble through a liquid/liquid interface. R. Bonhomme

Contraintes sur la stratégie de portage GPU

Code de calcul développé par une quinzaine de personnes : doctorants et chercheurs aux profils plutôt physiciens :

- des compétences HPC variables
- pas de compétences GPU
- code qui doit rester suffisamment « simple »
- garder une portabilité sur les architectures CPU

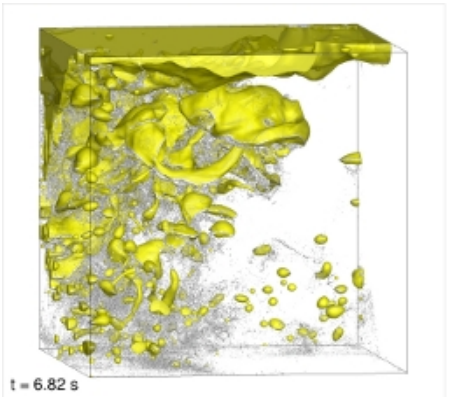
=> pour ces raisons, on a fait le choix de ne pas coder en CUDA



Arbre de développement Gitlab du code JADIM

Différentes phases du portage de JADIM sur GPU

Décembre 2016
définition d'un cas de calcul JADIM pour la phase de bench du nouveau super-calculateur de CALMIP



Mars 2017
benchs avec les constructeurs candidats à l'AO
1ere stratégies de portage sur GPU de la résolution du système linéaire

Décembre 2018
Participation au Hackathon GENCI
Branchement d'AmgX dans JADIM

Avril 2019
Hackathon GPU
Jülich
OpenACC
Managed Memory

Eté 2019
JADIM sur GPU
MPI Cuda-Aware

Benchmark supercalculateur CALMIP (2017)

2 constructeurs ont proposé une stratégie de portage pour la résolution du système linéaire :

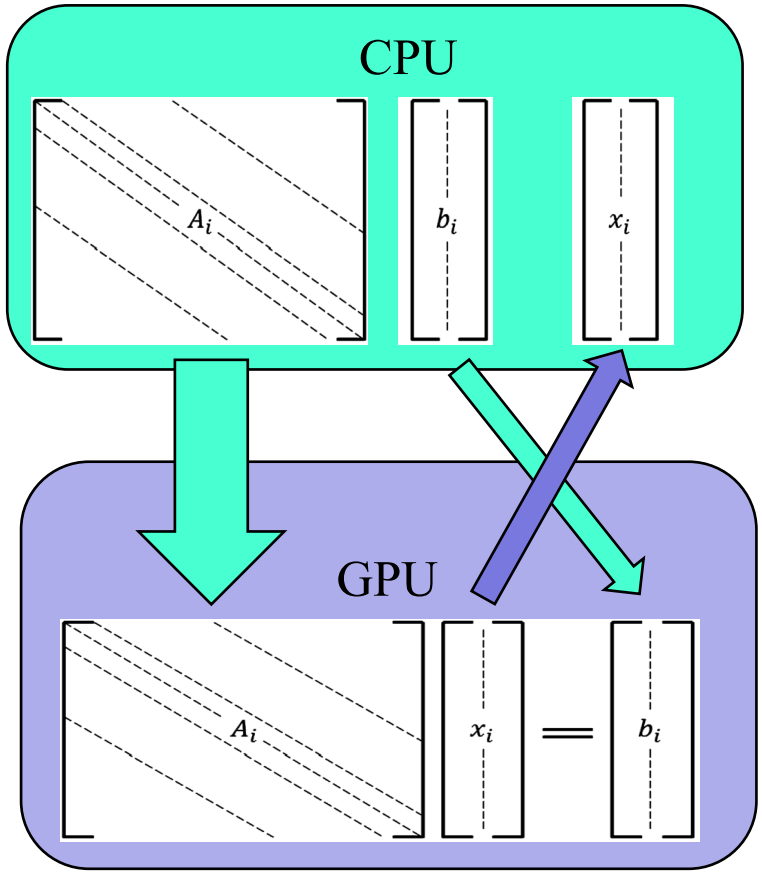
- PETSc avec support Cuda (seulement 5% du temps de résolution sur GPU)
- démonstrateur AmgX (démonstrateur avec un système de Poisson 2D)

Simulations multiphasiques :

- nombreux pas de temps pour simuler les processus physiques
- les coefficients de la matrice changent à chaque itération temporelle

$\Rightarrow$ transferts de données, un frein à un apport efficace des cartes accélératrices?

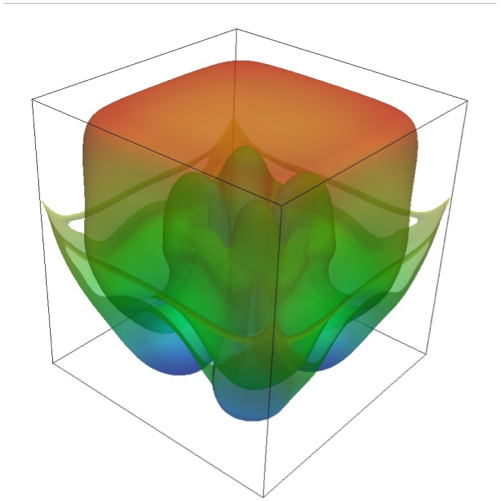
Quels gains possibles d'un portage de la résolution du système linéaire sur GPU?



Hackathon GENCI (décembre 2018)

Test de la bibliothèque AmgX (Algebric multigrid solver) développée par Nvidia, version Open Source pour résoudre les systèmes linéaires dans JADIM

- Etude de la faisabilité et du gain de performance vis-à-vis d'une solution CPU
- Etude de trois différentes tailles de problème (problème 3D : maillages de taille 100^3, 200^3 et 400^3)
- Calcul sur plusieurs itérations temporelles : reproduction d'une chaîne réelle de traitement
- Comparaison des temps de calcul par rapport à la solution CPU actuellement utilisée dans ce contexte (PETSc) avec un algo similaire : gradient conjugué préconditionné (JCG)



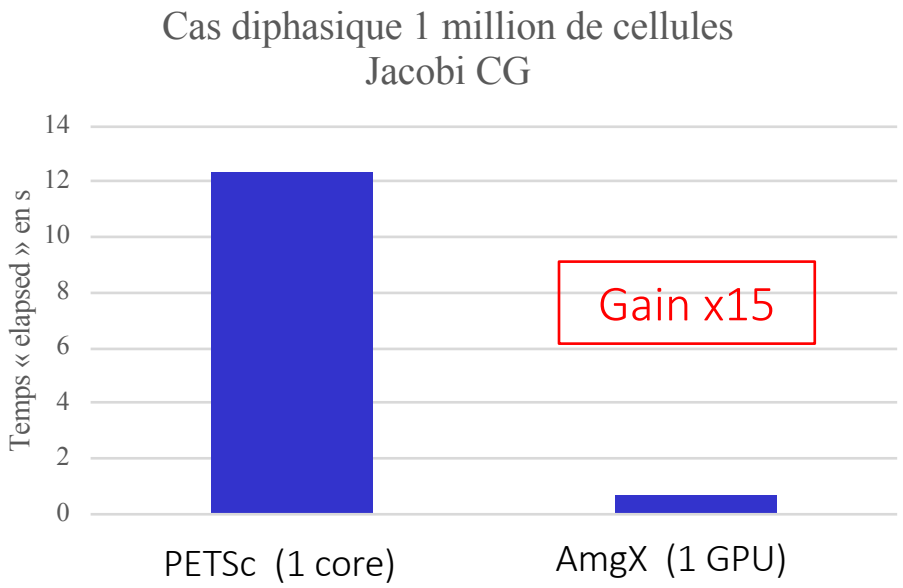
Crustal polydiapirs
(Convection dans la croûte terrestre)
A. Louis-Napoléon

Résultats Hackathon GENCI

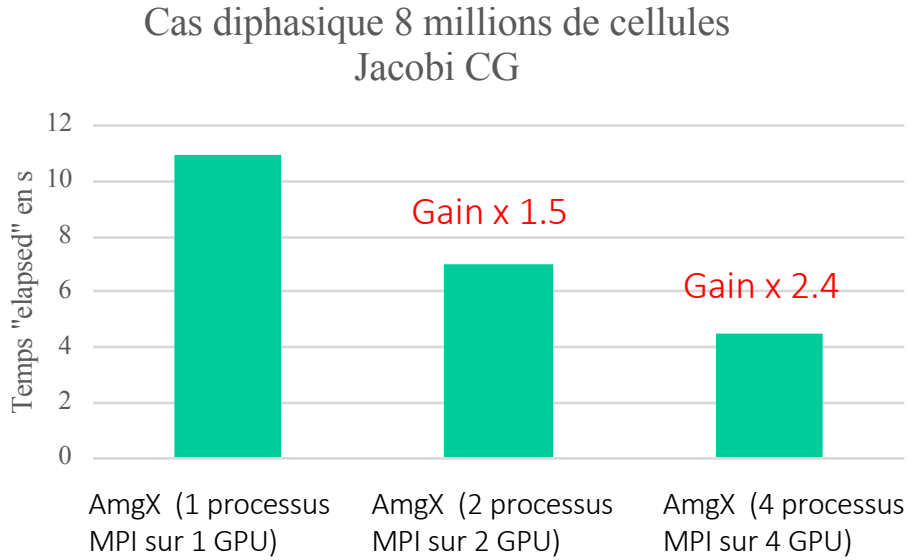
Démonstration de faisabilité pour :

- calcul séquentiel mono-GPU (1 processus MPI - 1 GPU)
- calcul parallèle mono-GPU (2 ou 4 processus MPI - 1 GPU)
- calcul parallèle multi-GPU (1 processus MPI par GPU)

Temps de transfert de la matrice du CPU vers le GPU < 10% du temps de résolution



Calcul séquentiel mono GPU



Calcul parallèle multi-GPU

Hackathon GPU (Jülich avril 2019)

"GPU Hackathon" organisé par OakRidge National Laboratory, NVIDIA et Super Computing Jülich
objectif : tester la possibilité de porter du reste du code sur GPU

Utilisation de OpenACC Managed Memory de PGI pour simplifier le travail de portage.

Mise à disposition d'un mentor Nvidia pendant une semaine

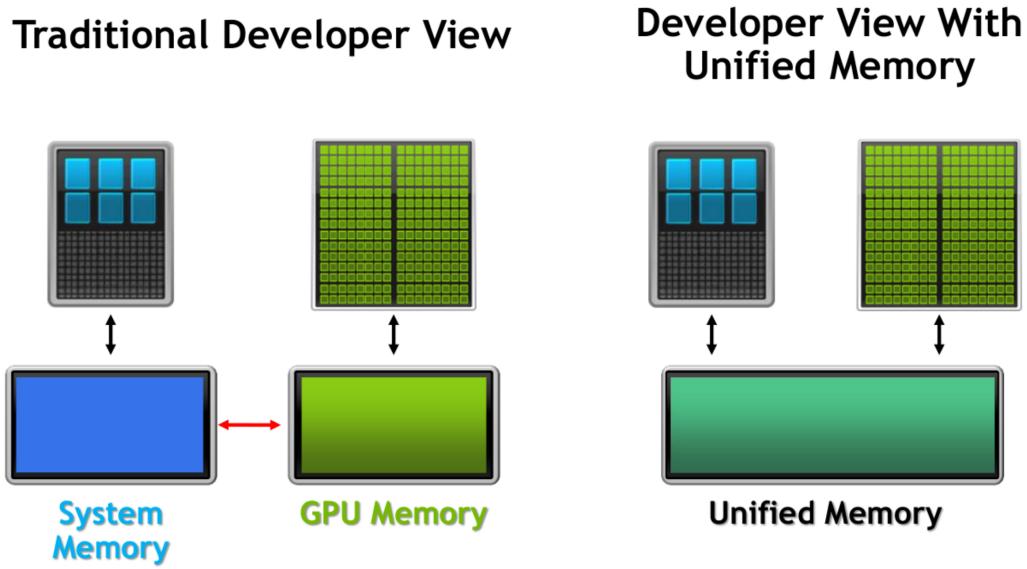


Schéma Mémoire Unifiée (Nvidia®)
UVA : Unified Virtual Adressing

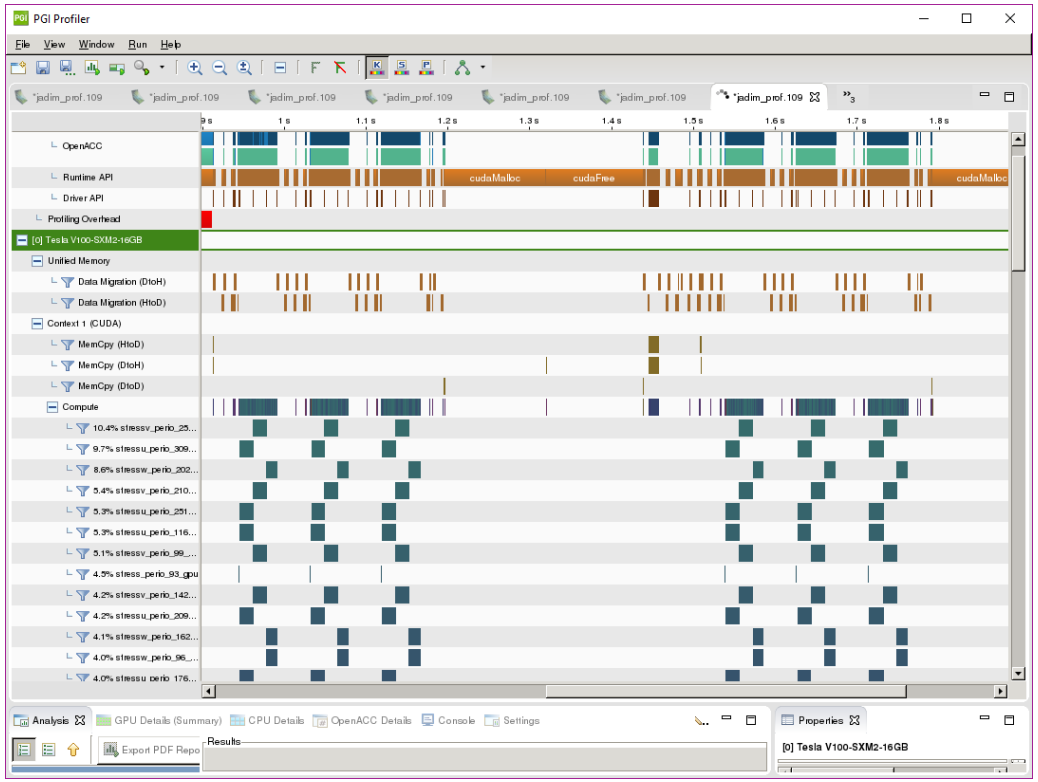
Hackathon GPU (Jülich avril 2019)

- Analyse des runs dans pgprof pour détecter les « CPU Page Fault »

- Pour évaluer le gain, il faut porter l'intégralité du code exécuté sur GPU

⇒ mettre des directives dans tous les fichiers sources utilisés

- Optimisation impossible pendant le temps imparti pour un cas de simulation avec une physique complexe.



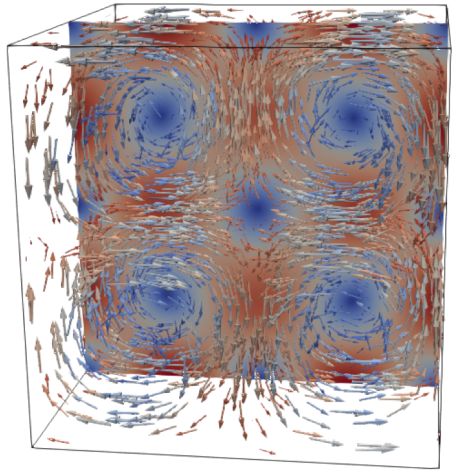
Simplification pour étudier la faisabilité

Nouveau cas de calcul monophasique avec des conditions aux limites périodiques pour réduire le nombre de fichiers sources utilisés et éviter les « hot spots » difficiles à optimiser en une semaine :

Avancement en temps de la solution :

- ~~Transport multi-phasique~~
- Advection-diffusion (Runge-Kutta 3)
  - termes explicites d'advection-diffusion
  - ~~partie semi-implicite pour les termes visqueux~~

  ~~Schéma de Crank-Nicolson (solveur type ADI)~~
- Méthode de projection (résolution système linéaire)



Taylor-Green 3D monophasique

Modification du code (1)

100 directives OpenACC ajoutées :

La plupart du temps, PGI se « débrouille » seul :

- 30 !$acc kernels
- 30 !$acc end kernels

Si besoin, il faut « aider » :

- 9 !$acc loop
- 16 !$acc parallel loop
- 9 !$acc parallel loop collapse()
- 3 !$acc parallel loop reduction()
- 3 !$acc parallel loop collapse () reduction()

```
!$acc kernels
do k = sk,ek
  do j = sj,ej
    do i = si,ei
      divp(i,j,k) = 0.
    end do
  end do
end do
!$acc end kernels

!$acc parallel loop
do k = sk,ek
  do j = sj,ej
    do i = si,ei
      ind = (ei-si+1)*(ej-sj+1)*(k-sk) &
          + (ei-si+1)*(j-sj) + i-si
      ph(i,j,k) = sol(ind)
    end do
  end do
end do
```

Modification du code (2)

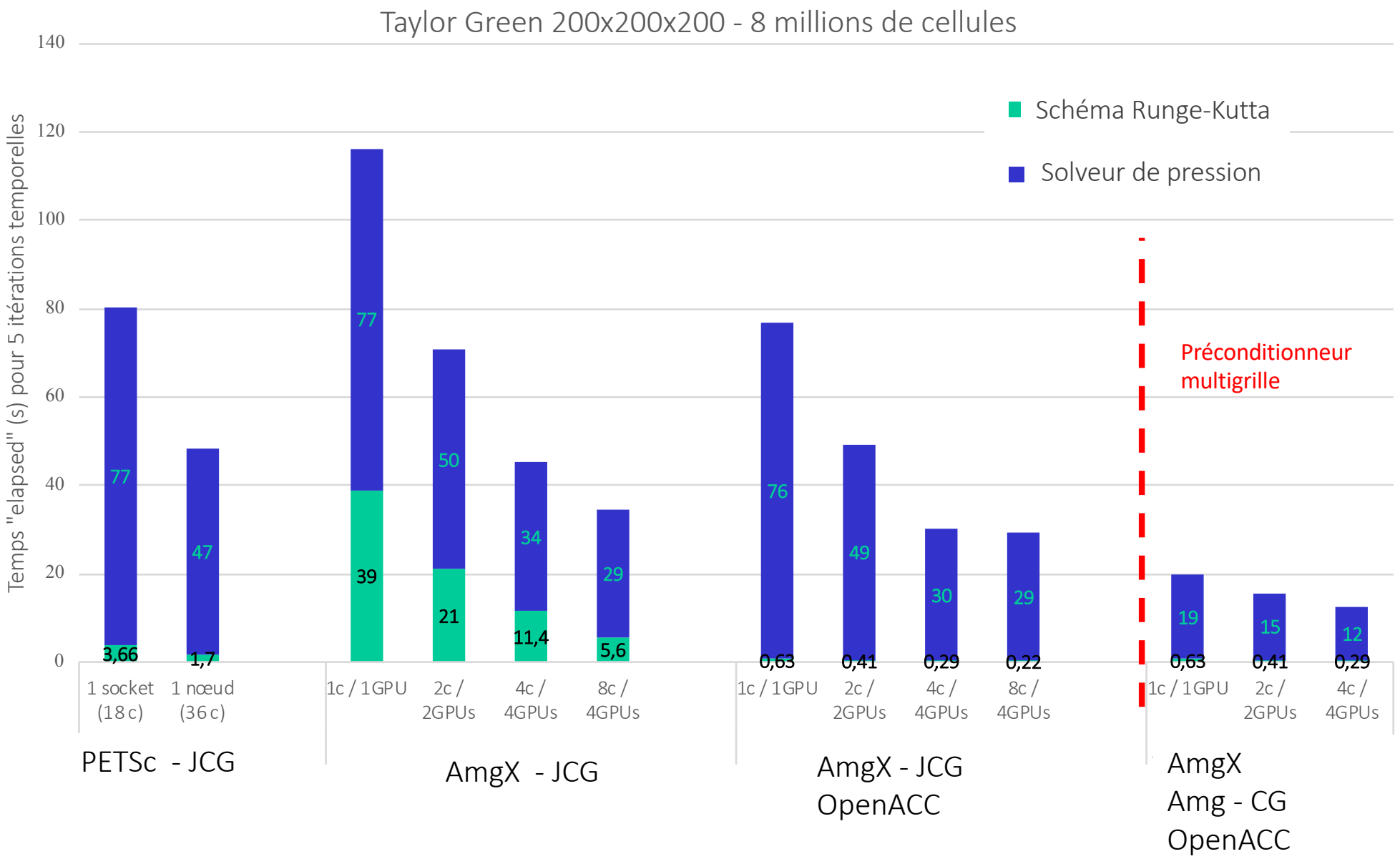
- Transformation des tableaux de structures en simples tableaux
- Suppression des index implicites dans les boucles
- Eclatement des boucles pour traiter l'intérieur du domaine d'un côté et les conditions aux limites de l'autre (suppression des tests dans les « grosses » boucles)

=> Ressemble à un travail d'optimisation vectoriel. Le code modifié tournera également plus vite sur CPU.

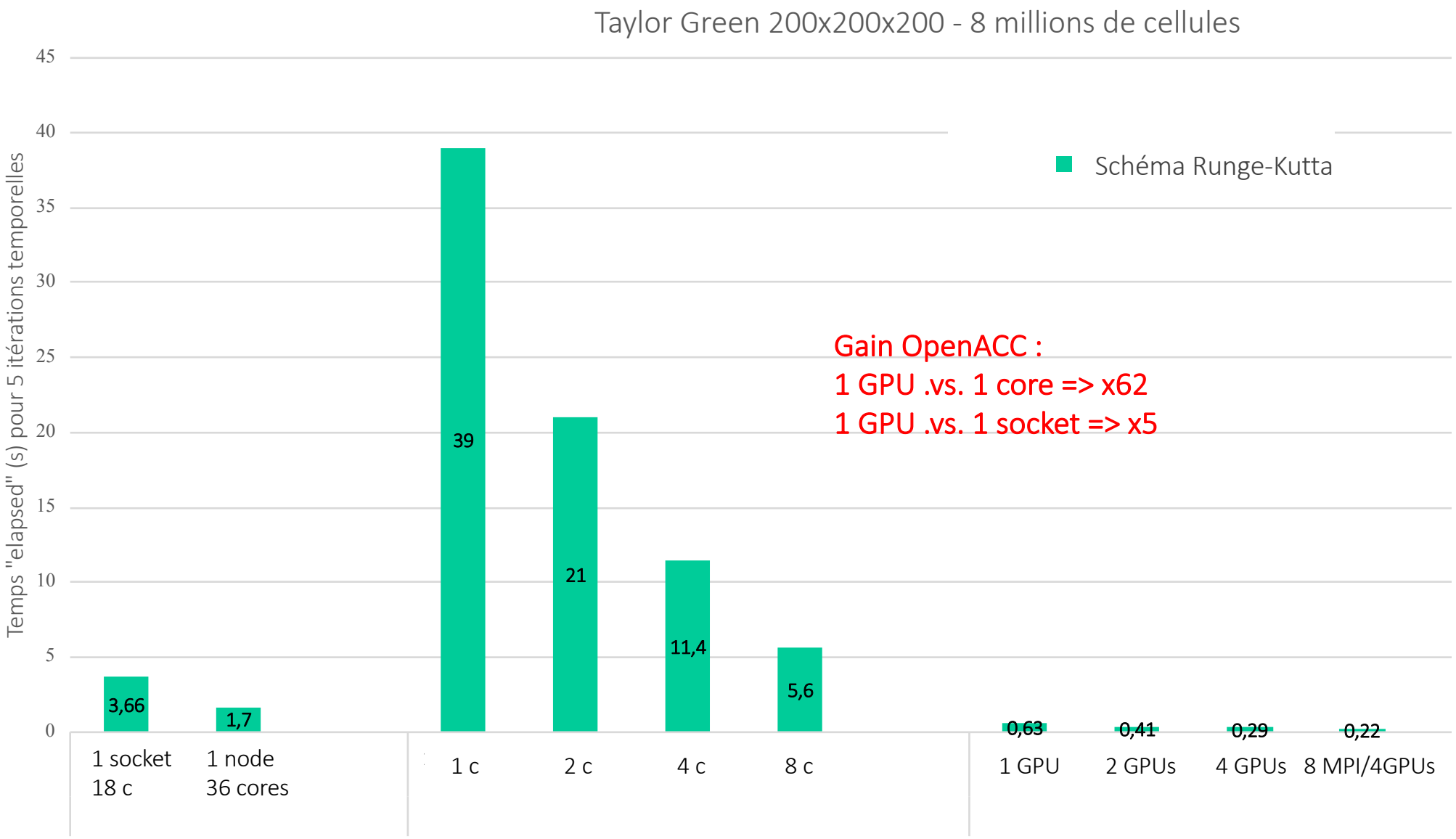
```
do k = sk,ek
  do j = sj,ej
    do i = si,ei
      ph(i,j,k) = sol(ind)
      ind = ind+1
    end do
  end do
end do

do k = sk,ek
  do j = sj,ej
    do i = si,ei
      ind = (ei-si+1)*(ej-sj+1)*(k-sk) &
          + (ei-si+1)*(j-sj) + i-si
      ph(i,j,k) = sol(ind)
    end do
  end do
end do
```

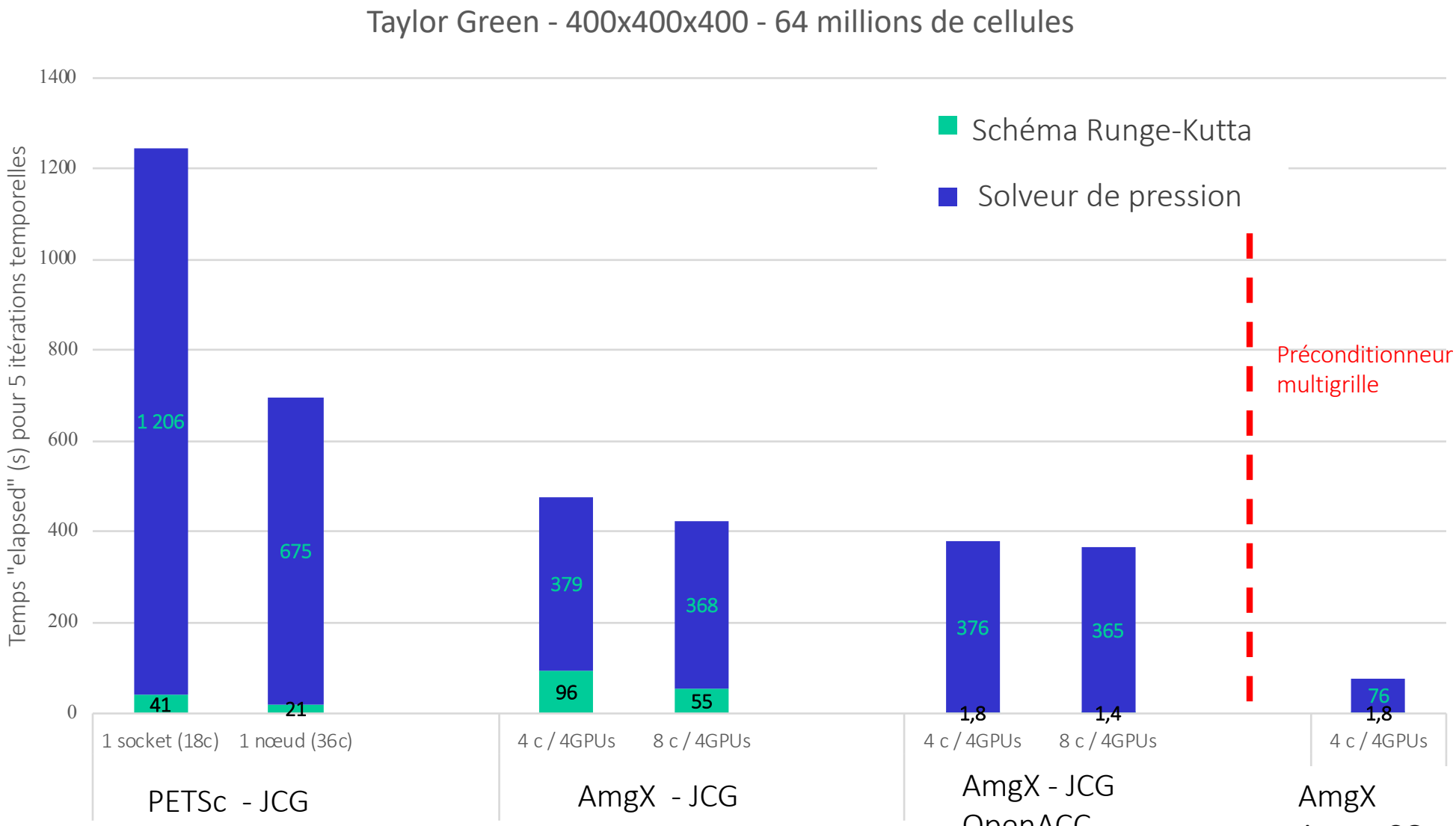
Temps de restitution total (cas 200x200x200)



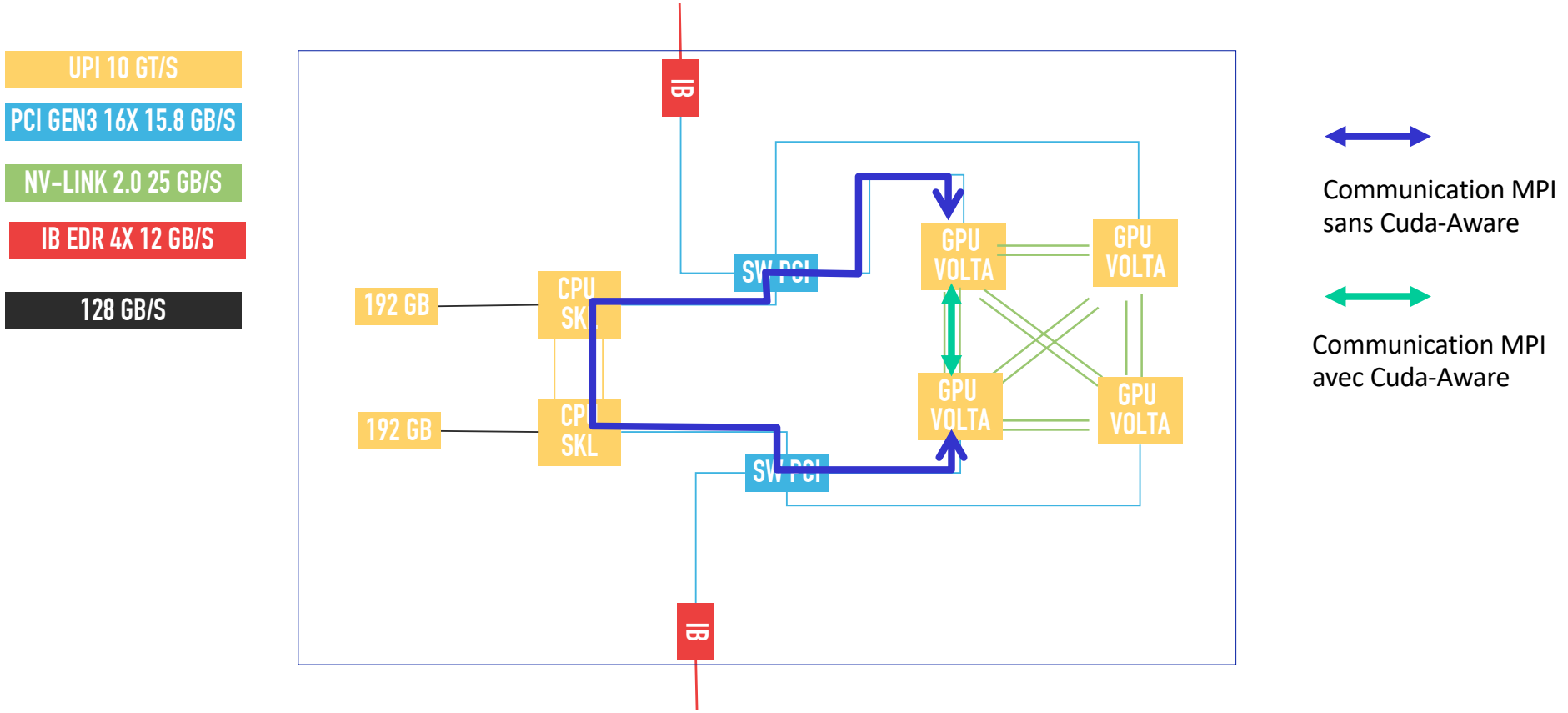
Temps de restitution partie OpenACC (cas 200x200x200)



Temps de restitution total (cas 400x400x400)



Tests MPI-Cuda Aware

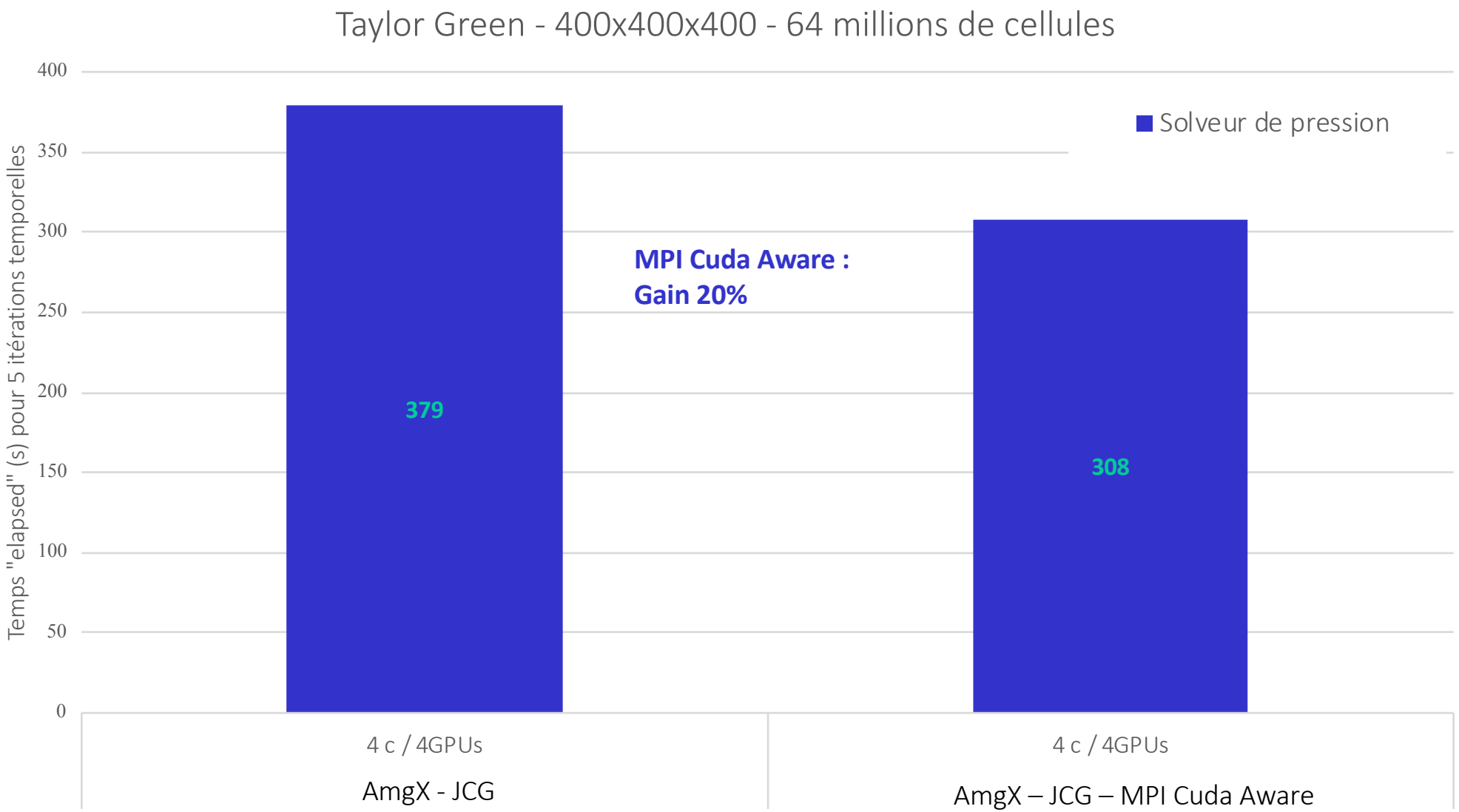


Communication MPI sans Cuda-Aware

Communication MPI avec Cuda-Aware

Architecture d'un nœud Volta d'Olympe

Temps de restitution AmgX (cas 400x400x400)



Difficultés rencontrées

- Impossibilité d'utiliser simultanément Cuda Aware et OpenACC Managed Memory
- Installation d'un environnement matériel et logiciel compatible : Versions de Cuda, du pilote graphique, de PGI, d'OpenMPI, …
- Placement des processus sur les GPUs, nombre optimal de processus MPI et de GPUs en fonction de la taille du problème.

Conclusion

- AmgX est une solution mature pour l'accélération de la résolution des systèmes linéaires creux dans JADIM
- OpenACC est prometteur avec des gains très intéressants sur les parties portées et optimisées du code
- Les 2 hackathons ont été des leviers importants pour le portage du code => rassemblement des « forces vives » sur un laps de temps dédié
- Collaboration étroite entre une doctorante et deux équipes techniques (IMFT et CALMIP) => montée en compétences de toute l'équipe.

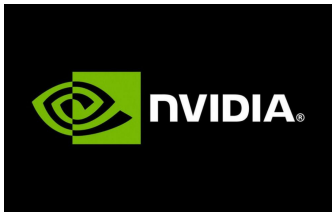
Perspectives

L'ensemble du travail de portage s'est concentré sur le cœur du code JADIM.

- étendre les directives aux différents modules du code et réaliser un travail d'optimisation pour des calculs de production
- utiliser OpenACC sans Memory Managed pour permettre la portabilité

  $\Rightarrow$ Allocations et copies des variables CPU $\Leftrightarrow$ GPU, quid de la maintenabilité du code?
- tester un solveur « CUDA batch » pour optimiser la résolution semi-implicite des termes visqueux (solveur ADI)
- appliquer les techniques et stratégies mises en œuvre à d'autres codes de mécanique des fluides incompressibles et/ou utilisant des méthodes implicites.

Remerciements

Ce travail n'aurait pas été possible sans l'aide de :

- François Courteille (Nvidia) pour ses premiers tests d'AmgX
- GENCI pour l'organisation du Hackathon qui nous a mis sur les rails
- Frédéric Pariente (Nvidia) pour nous avoir incités à faire le voyage à Jülich
- OAK RIDGE & Jülich Super Computer
- Andreas Herten et Kaveh Haghighi Mood (Jülich Super Computer)
- Jiri Kraus, notre « mentor » Nvidia à Jülich
- Christophe Marteau et Emmanuel Courcelle (CALMIP) pour l'installation d'un environnement Cuda / PGI / OpenMPI /Driver Grahique compatible
- l'IMFT pour le financement du déplacement en Allemagne