Portage de Méso-NH sur Super-Calculateur Hybride Avec OpenACC Sur GPU NVIDIA et AMD

J.Escobar(1),P.Wautelet(1),N.Alaoui(2),P.Vezolle(3),P.E.Bernard(3) ;

(1) LAERO , (2) Eolen , (3) HPE

Le Modèle de Recherche Méso-NH

- Un développement conjoint CNRS/ Météo-France
  - Modèle Non Hydrostatique pour traiter une vaste gamme de phénomènes atmosphériques de 1000 km au mètre
  - jeu complet de paramétrisations physiques, dont nuages, turbulence et rayonnement
  - couplé au modèle de surface SURFEX
  - configuration en cas idéalisés 1D, 2D, 3D et cas réel avec capacité d'imbrication pour descente en échelle
  - chimie et aérosols en phase gazeuse et aqueuse
  - bilan, traceurs, opérateurs d'observation (sat, radar, GPS)

- Parallélisation
  - F90 + MPI = 1 million de lignes de codes
  - 100% vectoriel <=> ARRAY SYNTAX (presque pas de boucle)
  - Décomposition de Domaine 2D X*Y , Z complet
  - Point Difficile, Solveur de Pression <=> Équation Elliptique à inverser
    - Pré-conditionneur « FFT3D » + Méthode Gradient Conjugué

- WEB : http://mesonh.aero.obs-mip.fr (/mesonh55/Download)
  - Version courante sous git/lfs
  - PACK-MNH-V5-5-0 : Licence OpenSource : CECIL-C

Toute version reproductible au bit près en parallèle

Architecture Massivement Parallèle

CLASSIQUE <=> HIER

- **ANL(USA)/MIRA/IBM-BG/Q**
  - **49 152 nœuds * 16 cores** = **10 PFLOPS**

HYBRIDE <=> AUJOURD'HUI

- RIKEN(JAPIN)/FUGAKU :
  - 158 976 nœuds * A64FX = 537 PFLOPS
- ORNL(USA)/SUMMIT/IBM :
  - 4 608 nœuds * 6 NVIDIA/V100 = 200 PFLOPS
- NSCC(CH)/SUNWAY/NRCPC :
  - 40 960 nœuds * SW26010 = 125 PFLOPS

EXASCALE <=> AVANT-HIER ;-)

- ORNL(USA) FRONTIER
  - ~10 000 nœuds Cpu AMD + GPU AMD Instinct = 1.5 EF (No1 T500)
- CSC(Finland/EuroHPC JU) LUMI-G
  - 2560 nœuds Cpu AMD + GPU AMD Instinct = 0.5 EF (No3 T500)
- CINES(France/GENCI) ADASTRA
  - 360 nœuds Cpu AMD + GPU AMD Instinct = 0.07 EF (No10 T500)

EXASCALE <=> DEMAIN ?

- USA AURORA = 1EF : ~10 000 nœuds Cpu Intel + ~~Xeon-PHI~~ → GPU Intel Xe Architecture
- EUROPE MONT-BLANC : Cpu ARM + Accélérateur ARM+RISC-V(+FPGA)

Méso-NH + GPU/Accélérateur
La seule voie pour l'ExaFlops …

- Quelle Architecture /Compilateur / Langage ?

  NVIDIA : CUDA avec OpenACC

  AMD : ROCM + OpenMP / Cray + OpenACC

  Intel : OneAPI + OpenMP

  ARM : ???

- Pour Méso-NH , portage avec OpenACC ...

  - commencé dès 2010 sur PC + GPU Nvidia et cluster Labo NUWA
    avec les premières versions du compilateur PGI+ACC et de CUDA .
  - poursuivi sur différents prototypes GENCI dont OUESSANT/IDRIS
    Cpu IBM-PW8 + GPU Nvidia P100 + lien nvlink rapide CPU-GPU
  - et puis sur premier Super-Calculateur Hybride : Jean-Zay à l'IDRIS
    Cpu Intel + GPU Nvidia V100
  - et aussi au travers de PRACE sur Calculateur Européen : MARCONO100
    Cpu IBM-PW9 + GPU Nvidia V100 + lien nvlink rapide CPU-GPU

Méso-NH + GPU(NVIDIA) : OpenACC

- Parallélisation facile

  - l' « array syntax » parallélise & vectorise toujours, commentaire dans le code

```
!$acc kernels
 A = B + C
!$acc end kernels
```

  - … mais optimisation difficile
    <=> éviter/gérer/optimiser les copies de données CPU ↔ GPU ↔ Réseau

```
!$acc data present/copy/update(in/out)
      ….
!$acc end data
```

- Beaucoup, beaucoup, beaucoup de BUGs de compilateur (PGI/NVIDIA)
- Restructuration du code pour l'optimisation/reproductibilité des calculs :
  - Pas d'appel de fonctions retournant des tableaux
    (réécriture en subroutine + tableau temporaire)
  - Pas de tableaux automatiques, sinon on passe son temps à allouer/désallouer des tableaux <=>
    remplacé par des « piles de tableaux » pré-alloués + pointer fortran
  - Optimisation échanges de halos , GPU-direct + communications asynchrones
  - Recodage des fonctions « log;exp,… » pour Bit-reproductibilité
  - Problème Bug/Perf compilateur , certains « array syntax/where » → DO CONCURRENT

Méso-NH + GPU(NVIDIA) : OpenACC

- Qu'est-ce qui a été porté avec OpenACC?
  - seulement les routines les plus consommatrice
    5 % du code , mais 90 % a 95 % de temps de calcul
- Dynamique
  - Schéma d'advection PPM , pour les variables météorologiques (eau, glace, etc …)
  - Schéma d'advection WENO , pour les variables de vents (UVW)
  - Solveur de pression : pour les GPU passage FFT 3D→ MultiGrille (cf slide suivante)
- Physique
  - Schéma de Turbulence 3D (TKE)
  - Schéma de Nuages (Ice3)

- Méso-NH est bit-reproductible entre exécution sur multi-CPU et multi-GPU , comment fait-on ?
  - Développement/Utilisation de 2 « outils » intégrés dans le code
    - Utilisation d'une Librairie « Bit-reproductibilité » CPU/GPU(OpenACC) pour des fonctions « log;exp,… »
      C++ + OpenACC + interface F90 ELEMENTAL
      (et les bonnes options de compilation, ieee , -nofma , etc …)

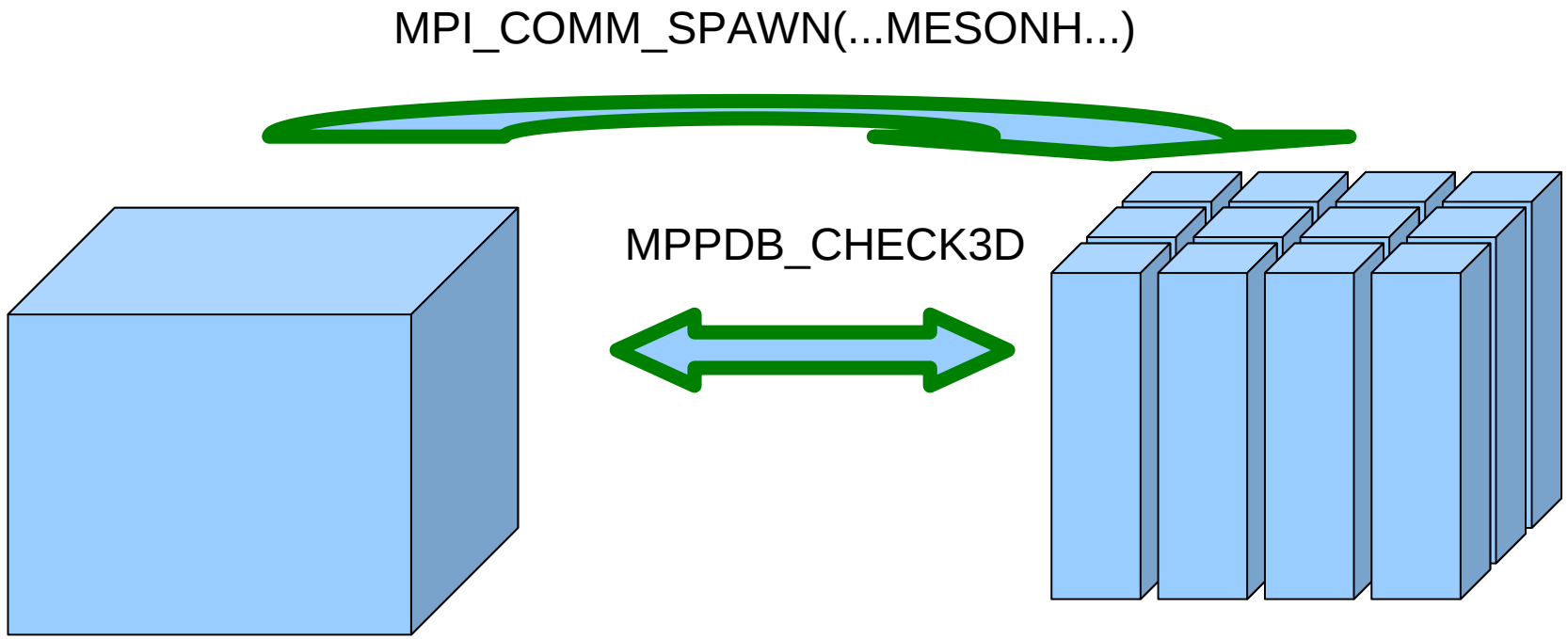
    http://mesonh.aero.obs-mip.fr/gitweb/?p=MNH-git_open_source-lfs.git;a=tree;f=src/LIB/BITREP;hb=refs/heads/MNH-55X-dev-OPENACC

    - « MPPDB_CHECK » , pour vérifier les valeurs des tableaux entre exécution sur CPU et GPU , à la volée

Méso-NH + GPU(NVIDIA) : OpenACC

« MPPDB_CHECK » , pour vérifier les valeurs des tableaux entre exécution sur CPU et GPU , à la volée

- Le **même code** (compiler avec -acc=host,gpu) est exécuté sur CPU et GPU
- Le maître lance l'esclave avec « mpi_comm_spawn »
  - Le maître tourne sur le CPU avec ACC_DEVICE_TYPE=HOST
  - L'esclave tourne sur le(s) GPU(s) ACC_DEVICE_TYPE=NVIDIA

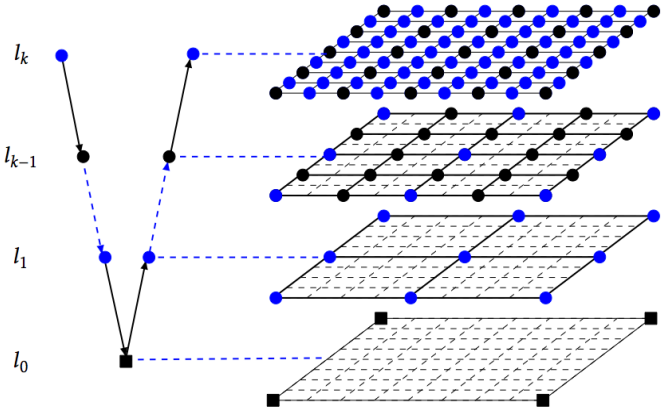


Solveur de Pression MNH FFT3D => Solveur Multi-grille Géométrique

- Pourquoi FLAT_INV à base de « FFT 3D » ne passera pas sur les GPU = Tesla V100?
  - Bande passante Mémoire 900 GB/sec
  - Bande passante intra-noeud Nvlink = 300 GB/sec
  - Bande passante inter-noeud PCI-3 = 32 GB/sec

  => transposition 30 fois plus lente
  que les accès mémoire (& calculs)

  => Besoin d'une méthode « plus locale »
- Solveur Multi-grille Géométrique « TensorProductMultiGrid » version CPU Fortran
  - Eike Müller, University of Bath (art. ref. juillet 2014):
    - MultiGrille Géométrique
      - avec Raffinement Horizontale
      - & Relaxation en Ligne Verticale »
    - adapté à la forte anisotropie en Z des modèles Météo
      (développement fait pour le UK Met Office)
    - Une dizaine de sources fortran90+MPI ,
- Interfacé/modifié pour Méso-NH
  - Testé jusqu'à 64k cores sur CPU
  - Porté sur GPU avec OpenACC
    - Pour faciliter le portage on a utilisé la « Managed-Memory »



Méso-NH GPU(NVIDIA) : Jean-Zay Intel+GPU-V100 Marconi100 PW9+Nvidia GPU-V100

- Résultats pour Méso-NH complet
  - Code compilé en OpenACC + Managed Memory (pour le solveur MG)
  - Testé sur le cas « Hector le Convector » , Grille 1024x1024x129pts pour la version de Méso-NH MNH-544 + « modif OpenACC » + « librairie math bit-reproductible entre CPU et GPU » .

- Le Solveur « Multi-Grille » utilisant forcément 2^(2*N) processeurs , soit pour ces tests :1, 4, 16, 64 ou 256 tâches MPI , on doit choisir souvent un nombre de tâches MPI inférieur au maximum autorisé :

  - 32 cores + 4GPU par nœud pour MARCONI100(CINECA)
  - 40 cores + 4GPU par nœud pour Jean-Zay(IDRIS)

- De plus chaque nœud n'ayant que 4 GPU, on utilisera le MPS (Multi-Process Service) pour surbooker « au mieux » ce GPU et arriver au nombre de tâches requis par le solveur MG
  - c'est le paramètre NP(best) dans les tableaux à suivre .
    - Par exemple : à 2 nodes , soit 8 GPU , on peut utiliser soit 16 soit 64 tâches MPI .
    - La solution NP(best) = 16 tâches avec GPU est celle qui donne les meilleures performances et est donc reportée ici.

Méso-NH GPU(NVIDIA) : Jean-Zay Intel+GPU-V100 Marconi100 PW9+Nvidia GPU-V100

- Légende :
  - NN = nombre de nœuds utilisés
  - NP(Max) = nombre max de taches MPI ↔ nombre de cores disponibles
  - NP(best) = nombre de tâches MPI ayant donné les meilleures performances
  - Méso-NH/MG (temps)= temps de Méso-NH avec le nouveau solveur Multi-Grille
  - Méso-NH/FFT(temps)= temps de Méso-NH avec le solveur standard utilisant les FFT-3D

  - Pour les lignes SpeedUP
    - Méso-NH/FFT = SpeedUP du Solveur standard-FFT sur CPU versus le MG sur GPU

    - Méso-NH/MG = SpeedUP du Solveur Multi-Grille sur CPU versus GPU

Méso-NH GPU/NVIDIA : Marconi100 PW9(32c)+4xNvidia GPU-V100

MARCONI-100 / IBM-PW9 + 4*GPU-NVIDIA/V100(16GO)				
Hector 1024x1024x128pts 100 pas de temps		NN(GPU)		
		4(16)	8(32)	16(64)
	NP(Max)	128	256	512
Méso-NH/MG PW9+V100	NP(best)	16 (64 Low Mem)	64	256
	Gri.Local	256^2	128^2	64^2
	Time(sec)	1217.678	243.392	124.461
Méso-NH/FFT PW9	NP(best)	128	256	512
	Time(sec)	3485.194	1414.476	735.248
Méso-NH/FFT	SpeedUP	2.86	5.81	5.90

Méso-NH GPU/NVIDIA : Jean-Zay Intel-CLK(2*20c)+4xNvidia GPU-V100

Jean-Zay / Intel-CLK + 4*GPU-NVIDIA/V100(32GO)				
Hector 1024x1024x128pts 100 pas de temps		NN(GPU)		
		4(16)	8(32)	16(64)
	NP(Max)	160	320	640
Méso-NH/MG CLK+V100	NP(best)	64	64	256
	Gri.Local	128^2	128^2	64^2
	Time(sec)	622.520	345.560	186.107
Méso-NH/FFT CLK	NP(best)	160	320	640
	Time(sec)	3815.711	1780.178	926.098
Méso-NH/FFT	SpeedUP	6.12	5.15	4.98

Méso-NH GPU/NVIDIA : Marconi100 PW9(32c)+4xNvidia GPU-V100

MARCONI-100 / IBM-PW9 + 4*GPU-NVIDIA/V100(16GO)						
Méso-NH : Hector 1024x1024x128pts 100 pas de temps						
Algo	ADV.MET	ADV.UVW	TURB.TKEL	PRESSURE	CLOUD.ICE3	TOTAL
Méso-NH/ MG PW9+V100 Time(sec) 16NN/ 256NP(64GPU)	18.545	18.090	20.393	27.640	11.098	124.461
Méso-NHFFT PW9 Time(sec) 16NN/512NP	146.763	215.584	186.886	99.761	53.138	735.248
Méso-NH/ FFT SpeedUP	7.91	11.91	9.16	3.60	4.78	5.90

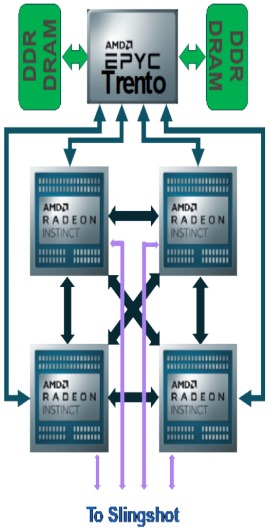
Méso-NH

What Else ;-)

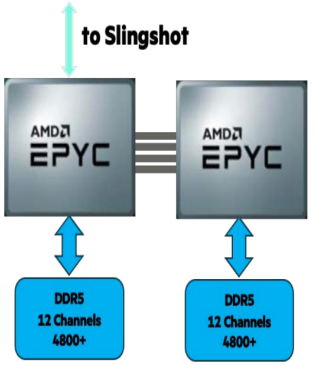
Méso-NH : Nouvelle Machine CINES ADASTRA HPE/CRAY Configuration Matériel

COMPUTE NODES ET BLADES

- 169 Bard Peak blades (2 nodes per blade)
  - 1x AMD Trento
  - 4x AMD MI200
  - 8x 32GB DIMMs
  - No on node storage
  - Quad injection Slingshot-11



- 134 Antero blades (4 nodes per blade)
  - 2x Genoa 320W 96c
  - 12x 32GB = 768 GB DDR5, 5200MHz
  - No on node storage
  - Single injection Slingshot-11



Méso-NH : Contrat de Progrès CINES - HPE

- Date de démarrage : **8 novembre 2021**
- Moyens humains :
  - Gouvernance technique : HPE
  - Ingénieurs expert HPC : 4 ETP* HPE + 1 ETP AMD
- Applications : MesoNH, TRUST, MagIC, GYSELA, MUMPS

+ Mise à disposition d'une plateforme de développement Apollo 6500 gen10 + (CPU Milan, GPU MI100)

+ Accès aux centres de compétences HPC

Méso-NH : Contrat de Progrès CINES – HPE
Logiciel : HPE CrayAMD

HPE CRAY PROGRAMMING ENVIRONMENT

- Cray Programming Environment
  - Cray Fortran Compiler
    - OpenACC 2.0 for Fortran.
    - OpenMP with offload for Fortran and C/C++
  - Clang (LLVM) C, C++, and UPC Compiler
  - CrayLibs (libraries and utilities)
  - All supporting documentation and man pages

Méso-NH + GPU (AMD): OpenACC

- Contrat de progrès :
  - Juan Escobar , Philippe Wautelet / LAERO
  - Naima Alaoui/Eolen, Pascal Vezolle,Pierre-Eric Bernard / HPE
  - Machine de portage CINES01 : 1 nœud 2*CPU'(*32c) Rome + 8GPU/AMD-MI100(lien PCI)

- Compilateur/Environnement F90 CRAY + OpenACC : Ce qui Marche ;-)

  - l' « array syntax » fonctionne avec OpenACC : OUF , pas besoin de réécrire tout le code !!!

  - MPPDB_CHECK : Mais il a fallu compiler/utiliser OpenMPI , car le MPICRAY ne supporte pas MPI_COMM_SPAWN (dans la version utilisée)

  - La Librairie « Bit-reproductibilité » : Mais il a fallu rajouter des directive OpenMP «omp declare target » dans le code C++ , car OpenACC n'est supporté que par le compilateur CRAY C++

Méso-NH + GPU (AMD): OpenACC

- Mais beaucoup de BUG de compilateur F90 CRAY pour OpenACC

  - Tout les problèmes ci-dessous soumis à HPE/CRAY, en attente de correction/amélioration du compilateur …

    … en attendant => Bypass

  - Plantage du compilateur : ICE ou SegFault à l'exécution

    - Problème avec : les chaînes de caractères , les variables scalaires allocatable , les % des types dérivés dans les parties calculs
    - Données « not found in present tab » / « host region overlap present » => Bypass :: allocatable → pointer + pile

  - Problème de performance à l'exécution

    - Tableaux « internal » , copié CPU<->GPU indûment => Bypass :: par macro CPP « present_cr », dupliquer les «!$acc … present(tab) »
    - «!$acc loop independent ... » , inhibe la parallélisation !!!! => Bypass :: macro CPP « acc_nv » , pour déactiver la ligne avec le compilateur Cray
    - Les routines ELEMENTAL « bit-reproductible », inhibe la parallélisation => Bypass :: array syntax/where → DO CONCURRENT
    - La plupart des calculs 3D (array syntax/ do nesté / do concurrent) ne collapsent pas en 1 seule boucle (le compilo NVIDIA collapse systématiquement et a de meilleurs performances)
    - Etc ...

Méso-NH + GPU (AMD): OpenACC
1 Noeud Prototype CINES01

- 1 Noeud prototype CINES01 :
  - 2 processeurs AMD Rome (2*32 cœurs x2 threads, 1To mémoire)
  - 8 accélérateurs AMD MI100 (1 AMD-MI100,32 GO HBM2)
  - Liens CPU<->GPU et GPU<->GPU «PCX»

  REM : dans le run suivant seuls 4GPUs sont utilisés car plusieurs étaient en panne

… résultats très très préliminaires (l'optimisation n'a pas encore commencé ...)

Méso-NH GPU AMD : Prototype CINES01 CPU/AMD-Rome(2*32c)+4xGPU-AMD/MI100

CINES01 / 2*AMD-Rome + 4*GPU-AMD/MI100(32GO)						
Méso-NH : Hector 512x512x128pts 100 pas de temps						
Algo	ADV.MET	ADV.UVW	TURB.TKEL	PRESSURE	CLOUD.ICE3	TOTAL
Méso-NH/MG Rome+MI100 Time(sec) 1NN/16NP(4GPU)	134.277	75.567	100.547	134.223	43.847	627.735
Méso-NN/FFT Rome Time(sec) 1NN/64NP	1063.979	665.709	521.207	374.908	209.323	3056.794
Méso-NH/FFT SpeedUP	7.92	8.80	5.18	2.79	4.77	4.87

Méso-NH + GPU (AMD): OpenACC
1 Noeud ADASTRA/GPU

- 1 Noeud ADASTRA/GPU :
  - 1 processeur AMD Trento (64 cœurs physiques x2 threads,256Go mémoire)
  - 4 accélérateurs AMD MI250x (1AMD-MI250X = 128 GO HBM2)
  - Liens CPU<->GPU et GPU<->GPU « Infinity Fabrics »
  - 4 connexions réseaux « Slingshot » à 200Gb/s chacune, connectées directement aux GPUs .

    REM : 1 AMD-MI250X, contient en fait 2 GPUs indépendants (± MI100)
    donc par nœud on voit 8 GPUs « utilisables »

- Premiers résultats préliminaires sur GPU-MI250X
  - Run réalisé par Pascal VEZOLLE de HPE, Merci ;-)
  - La partie CPU n'a pas été retournée ↔ reprise du run précédent sur prototype CINES01

Méso-NH GPU AMD : 1Noeud «ADASTRA/GPU» CPU/AMD-Rome(2*32c)<->4xGPU-AMD/MI250X(*2)

CINES01 / 2*AMD-Rome ↔ ADASTRA 4*GPU-AMD/MI250X(32GO)						
Méso-NH : Hector 512x512x128pts 100 pas de temps						
Algo	ADV.MET	ADV.UVW	TURB.TKEL	PRESSURE	CLOUD.ICE3	TOTAL
Méso-NH/MG Trento+MI250X Time(sec) 1NN/16NP(4GPU)	69.621	40.187	54.505	83.115	38.362	404.122
Méso-NN/FFT Rome Time(sec) 1NN/64NP	1063.979	665.709	521.207	374.908	209.323	3056.794
Méso-NH/FFT SpeedUP	15.28	16.56	9.56	4.51	5.45	7.56

Méso-NH sur GPU

CONCLUSION

- Une première version de Méso-NH intégrant le nouveau «Solveur Multi-grille» pour GPU en Fortran+OpenACC a été codée en Multi-GPU .
- Elle a été testée avec des bonnes performances sur 16 nœuds /64 GPUs , aussi bien sur architecture CPU-Intel que CPU-IBM + GPU Nvidia .
- Le SpeedUP ≥ 4.98 / CPU , tant que les tailles de grilles locales ≥ 64^2
- La version OpenACC est en cours (finalisation) de phasage avec la version officielle MNH-55X

- Un portage sur Architecture CPU+GPU AMD avec OpenACC en cours
  Premiers résultats encourageants à optimiser ...

→ Grand Challenge ADASTRA/GPU Méso-NH (octobre 2022) :

« Giga-LES de rafales avec Méso-NH »

- Grille 2048x2048pts / 100mètres , 64 noeuds / 256(x2) GPUs (voir plus ...)

- Contribution à l'ANR JCJCWINDGUST (https://anr.fr/Projet-ANR-21-CE01-0002)

Méso-NH

back-slide

Méso-NH + GPU : Filepp + MNH_Expand_Array

– MNH_Expand_Array :

- Un préprocesseur utilisant « filepp » <=> cpp like , amélioré programmable en Perl
- Pour convertir l'array syntax/where en loop nesté , do concurrent
  + directive OpenACC / OpenMP au besoin
- Ajout de commentaires dans le code + déclaration des bornes des boucles
  - Pour de l'array syntax

    !$mnh_expand_array(ii=iib:iie:iis , ij=ijb:ije:ijs , ik=ikb:ike:iks)

    Array syntax

    !$mnh_end_expand_array(Commentaire)

  - Pour les Where

    !$mnh_expand_where(ii=iib:iie:ip , ij=ijb:ije:ijs , ik=ikb:ike:iks)

    Where + Array Syntax

    !$mnh_end_expand_where(Commentaire)

https://github.com/JuanEscobarMunoz/MNH_Expand_Array

Méso-NH GPU : Jean-Zay Intel-CLK+4xNvidia GPU-V100

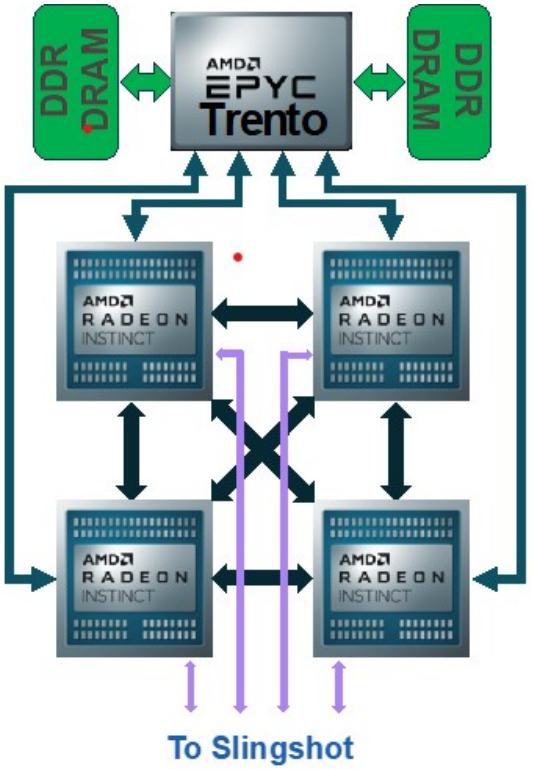
Jean-Zay / Intel-CLK + 4*GPU-NVIDIA/V100(32GO)						
Hector 512x512x128pts 100 pas de temps		NNODES(GPU)				
		1(4)	2(8)	4(16)	8(32)	16(64)
	NP(Max)	40	80	160	320	640
Méso-NH/MG CLK+V100	NP(best)	16	64	64	256	256
	Gri.Local	128^2	64^2	64^2	32^2	32^2
	Time(sec)	550.106	290.211	195.484	140.554	117.467
Méso-NH/MG CLK	NP(best)	TD	64	64	256	TD
	Time(sec)	TD	2412.452	1777.899	609.364	TD
Méso-NH/FFT CLK	NP(best)	40	80	160	320	640
	Time(sec)	4019.054	1945.792	1008.659	476.870	245.006
Méso-NH/MG	SpeedUp	TD	5.94	9.09	4.33	TD
Méso-NH/FFT	SpeedUP	7.30	6.70	5.15	3.39	2.08

Solveur MG Multi-GPU « StandAlone » : Marconi100PW9+4xNvidia GPU-V100

MARCONI-100 / IBM-PW9 + 4*GPU-NVIDIA/V100(16GO)						
MG 1024x1024x64pts 100 pas de temps		NNODES/NP(GPU)				
		1/1(1)	1/4(4)	1/16(4)	2/16(8)	4/64(16)
	NP(Max)	32	32	32	64	128
	Gri.Local	1024^2	512^2	256^2	256^2	128^2
MG PW9+V100	Time(sec)	12.48	4.466	5.567	3.799	4.407
MG PW9	Time(sec)	1252.0	253.3	64.91	59.18	15.84
Méso-NH/MG	SpeedUP	100.32	56.71	11.65	15.57	3.59

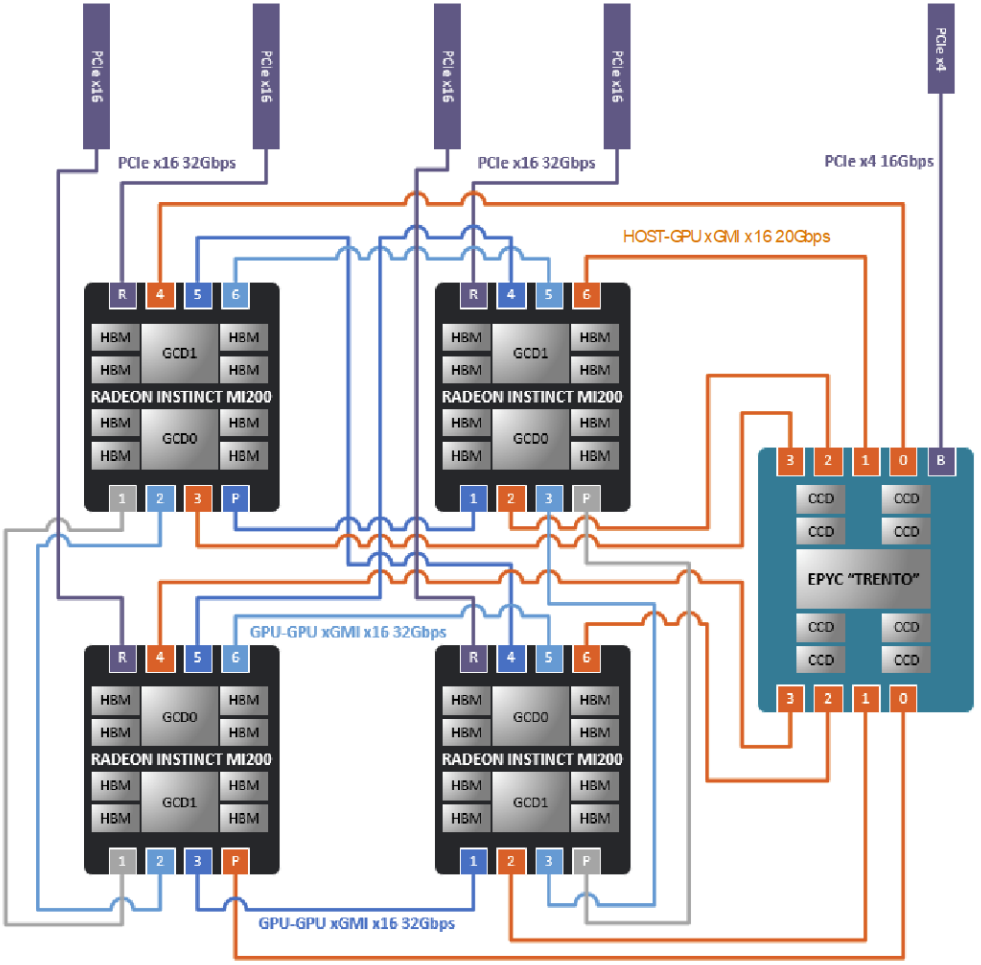
Méso-NH + GPU (AMD): OpenACC

– Noeud ADASTRA/GPU :



Méso-NH + GPU (AMD): OpenACC

– Noeud ADASTRA/GPU :



Méso-NH GPU : Marconi100 PW9+4xNvidia GPU-V100

MARCONI-100 / IBM-PW9 + 4*GPU-NVIDIA/V100(16GO)						
Hector 512x512x128pts 100 pas de temps		NNODES(GPU)				
		1	2(8)	4(16)	8(32)	16(64)
	NP(Max)	Low-MeM	64	128	256	512
Méso-NH/MG PW9+V100	NP(best)		16	64	64	256
	Gri.Local		128^2	64^2	64^2	32^2
	Time(sec)		245.724	125.98	89.663	70.718
Méso-NH/MG PW9	NP(best)		64	64	256	TD
	Time(sec)		1689.106	892.895	368.020	TD
Méso-NH/FFT PW9	NP(best)		64	128	256	512
	Time(sec)		1522.280	797.687	383.780	188.311
Méso-NH/MG	SpeedUp		6.87	7.08	4.27	TD
Méso-NH/FFT	SpeedUP		6.19	6.33	4.10	2.66

Méso-NH GPU : Marconi100 PW9+4xNvidia GPU-V100

MARCONI-100 / IBM-PW9 + 4*GPU-NVIDIA/V100(16GO)						
Méso-NH : Hector 512x512x128pts 100 pas de temps						
Algo	ADV.MET	ADV.UVW	TURB.TKEL	PRESSURE	CLOUD.ICE3	TOTAL
Méso-NH/MG PW9+V100 Time(sec) 4NN/ 64NP(16GPU)	19.350	17.465	19.659	25.296	10.466	125.980
Méso-NHFFT PW9 Time(sec) 4NN/128NP	185.906	214.645	188.279	78.738	55.525	797.687
Méso-NH/ FFT SpeedUP	9.60	12.29	9.57	3.11	5.30	6.33

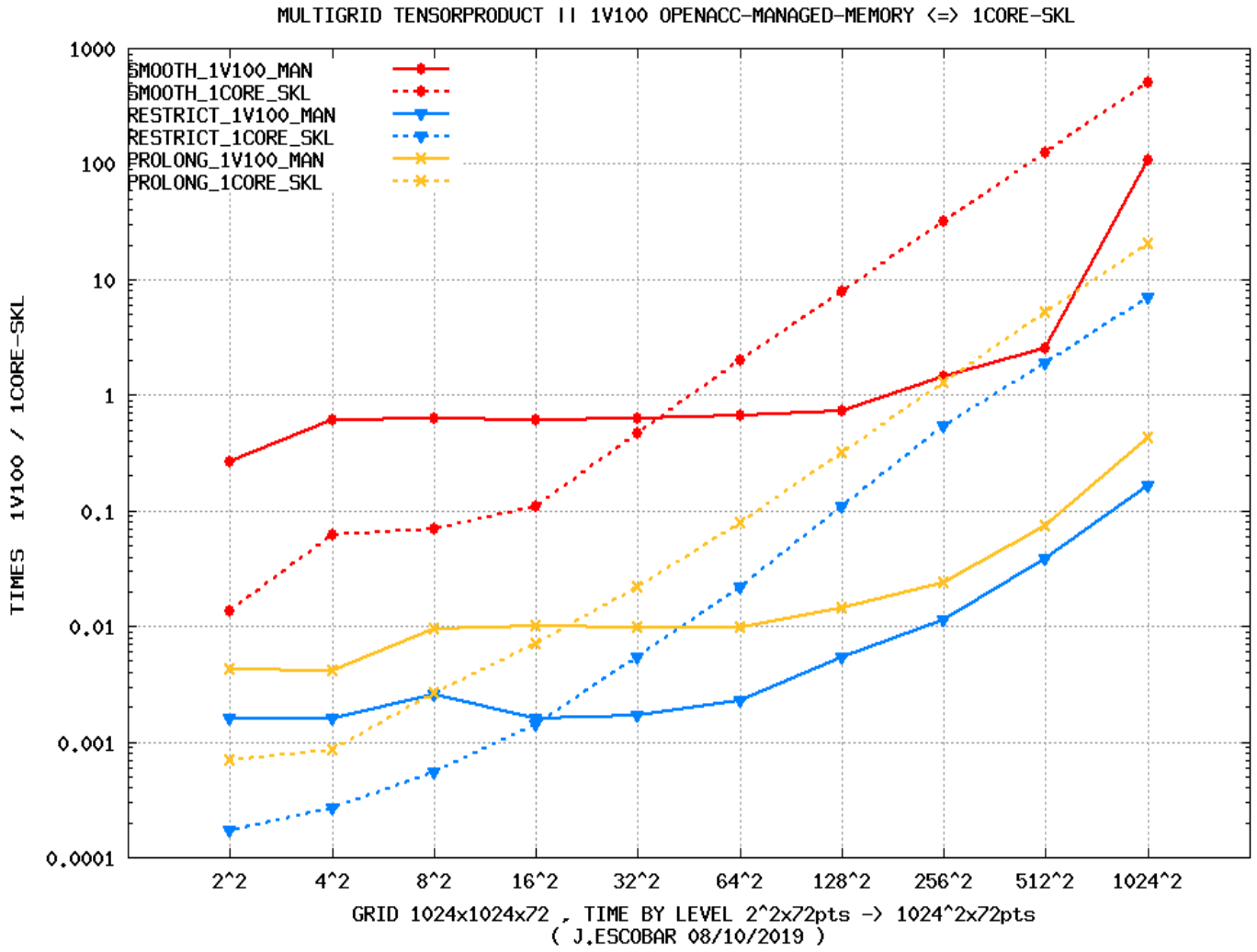
Méso-NH GPU : Jean-Zay(Prototype) Intel-SKY+8xNvidia GPU-A100

Jean-Zay / Intel-SKL + 8*GPU-NVIDIA/A100(40GO)			
Hector 512x512x128pts 100 pas de temps		NNODES(GPU)	
		1(8)	2(16)
	NP(Max)	48	96
Méso-NH/MG SKL+A100	NP(best)	16	64
	Gri.Local	128^2	64^2
	Time(sec)	354.419	227.060
Méso-NH/FFT SKL	NP(best)	40	80
	Time(sec)	4129.130	1963.850
Méso-NH/FFT	SpeedUP	11.65	8.64

Solveur Multigille Géométrique(Stand Alone)

Test préliminaire : Comparaison des temps de calculs

1GPU/V100 versus 1CORE-Sylake
Smoother/Prolongation/Restriction



Méso-NH GPU : Marconi100 PW9+Nvidia GPU-V100

- Nouveauté 2021 : Au travers d'un Accès Préparatoire PRACE nous avons continué nos efforts sur la machine MARCONI100(CINECA) (14ème au Top 500) , IBM-PW9+GPU-NVIDIA/V100 .
  - Poursuivre le portage du Solveur de Pression Multigrille sur GPU , en profitant de la « Managed Memory » , plus rapide sur ce type d'architecture « Cpu-IBM+GPU-Nvidia » possédant trois liens NVLINK entre les CPU et les GPU , et offrant du coup un débit de copie de mémoire-CPU à mémoire-GPU d'un ordre de grandeur supérieur aux architectures « Cpu-Intel+GPU-Nvidia » comme Jean-Zay connecté uniquement par PCI-Express .
- Plusieurs améliorations importantes on été rajoutées au code MG :
  - Remplacement de « tableaux automatiques » en « pointer contiguous » et toutes les allocations mémoire « allocate » par nos propres routines « mnh_allocate* »
  - Remplacement d'instruction « array syntaxe » en « do concurrent » (bypass bug nvfortran)
  - Allocation une fois pour toutes de tous les tableaux 3D « temporaires »intervenant dans le solveur MG
  - Rajout des directives «!$acc kernels + loop indepedent » aussi bien dans le code MG proprement dit (stand-alone), que dans la partie des routines Méso-NH appelant ce préconditionneur MG dans un solveur Conjugé Residual (pressurez , zsolver , zsolver_inv ,contrav , gdiv , etc …)
  - Modification des routines utilisant MPI dans le solveur MG , en passant par des buffers intermédiaires pré-alloués sur GPU (sans la Managed Memory) pour tirer profit des communications « GPU Direct»
  - Contournement d'un bug des compilateurs nvidia/21.X sur les «!$acc atomic » rendant les calculs complètement faux, mais uniquement sur CPU !
  - Pour les routines « purement » Méso-NH , remplacement des routines « update_halo » et « update_halo2 » par des routines optimisées pour les GPU get_halo_d et get_halo2_d elles aussi utilisant des buffers préalloués sur GPU et le « GPU direct »
  - Rajout d'un paramètre en namelist «iswitch_cpu_gpu » permettant de basculer les calculs du MG sur CPU quand la sous-grille locale est trop petite pour apporter un gain de performance sur GPU (ce paramètre sera à régler en fonction de la taille de grille initiale et du nombre de GPU/CPU)

Méso-NH Standard : Fujitsu AF64X

- Nouveauté 2021 :
  - Toujours grâce a la cellule de veille technologique, les premiers tests de portage ont été faits sur le nouveau prototype , Irene-ARM du TGCC , comprenant 80 nœuds ARM-A64FX <=> les processeurs vectoriels avec mémoire rapide HBM, équipant le Super Calculateur Fugaku Japonais , 1er au Top 500 , depuis sa mise en route en juin 2020 .
- Sur ce prototype nous avons testé 4 compilateurs :
  - gnu/gfortran/11.1.0
  - arm/armflang/21.0.0 (basé sur llvm 11.0.0)
  - nvidia/pgf90=nvfortran/21.5
  - fuji/frt/1.1.0
- Le code a pu être compilé avec ces 4 compilateurs , mais nous n'avons réussi à exécuter nos cas tests qu'avec les compilateurs gnu/gfortran et arm/armflang , les 2 autres compilateurs générant des erreurs mémoire « SegFault » impossibles à identifier/résoudre même avec le debugger « ddt » .
- De plus les performances avec le compilateur armflang ne scalent pas bien du tout , donc nous ne reporterons pas ici les performances avec ce compilateur mais seulement celles avec le compilateur gfortran .

Méso-NH + GPU/Accélérateur

EN FRANCE

- GENCI : 2016-21 Mise en place de plusieurs prototypes Accéléré
  - FRIOUL / CINES(MESRI) : 48 nœuds * Intel Xeon-PHI KNL * 68C
  - OUESSANT/ IDRIS(CNRS) : 12 nœuds * [2 IBM-PW8*20C + 4GPU Nvidia P100]
  - INTI/ TGCC(CEA) : 20 nœuds * 2*ARM-TX2*32C
  - IRENE-A64FX/TGCC(CEA) : 80 noeuds*1*Fujitsu A64FX*48C
  - JEAN-ZAY/IDRIS(CNRS) : 3 nœuds *[2I ntel SKL + 8GPU Nvidia A100]

- Machines de production (2021)
  - accéléré
    - IRENE-KNL / TGCC(CEA) : 828 nœuds * Intel Xeon-PHI KNL *68C = 2 PF
    - JEAN-ZAY / IDRIS(CNRS) : (261+351) nœuds * [2 Intel CLK*40C + 4GPU NvidiaV100] = 15 PF
  - classique
    - ~~OCCIGEN/CINES(MESRI)~~ : 1260 nœuds Intel BW + 2160 nœuds Intel HW = 3.5 PF
    - IRENE-AMD / TGCC(CEA) : 2292 nœuds * AMD-EPYC 2*64C = 12 PF
    - BELENOS+TARANIS/METEO-FRANCE : 2 * 2256 nœuds * AMD-EPYC 2*64C = 2*10 PF

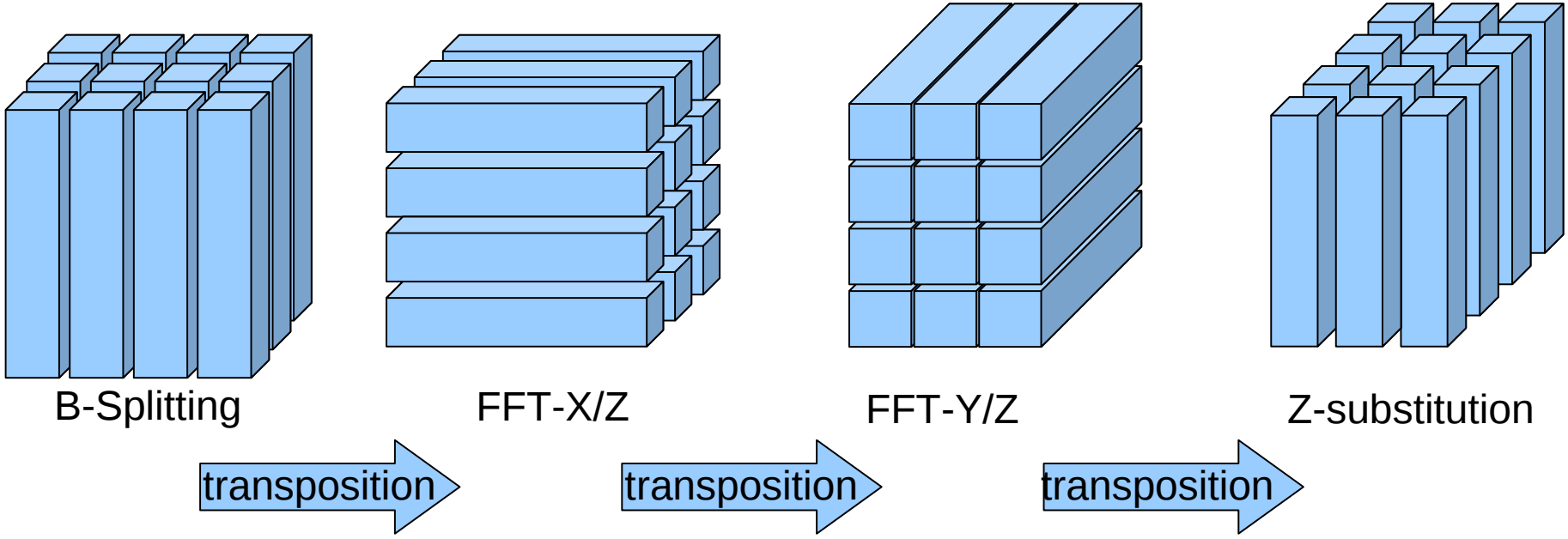
- Machines à Venir (2022)
  - ADASTRA / CINES(MESRI) : X nœuds AMD-EPYC++ * 4 GPU AMD MI250X = 75PF

Solveur de Pression MNH

Méthode itérative : « Gradient Conjugué » + Pré-conditionneur FLATINV

- Parallélisation du pré-conditionneur FLAT_INV :

  - décomposition de domaine dans 2 dimensions, à tour de rôle (Tetrix), avec transposition des données entre tous les processeurs à chaque étape FFT-X/Y ou Élimination de Gauss en Z



Solveur de Pression MNH
=> Solveur Multi-grille Géométrique

- Solveur Multi-grille Géométrique « TensorProductMultiGrid »

  - Adaptation à Méso-NH de la version Fortran90 :
    - Interface et appel a la place de FLAT_INV
    - Adaptation au équations de Méso-NH ↔ coefficients des « tensorproduct »
    - Codage des conditions aux limites propre à Méso-NH <-> Newmann
    - Debuggage du code parallèle «original» ↔ version bit-reproductible
    - Premier test préliminaire de Méso-NH sur Grande Grille sur IRENE(TGCC):
      - Pas mal de paramètres libres à régler: Méthode itérative (Richardson/CG), taux de convergence, Smoother(Jacobi, SOR …) , nombre de niveaux , Restriction, Prolongation,Solveur « gros grain »
      - Hector le Convector 2048x2048x256
        Temps Solveur, 1OO pas de temps : 256P/220.765s | 1024P/110.932s | 4096P/17.828s

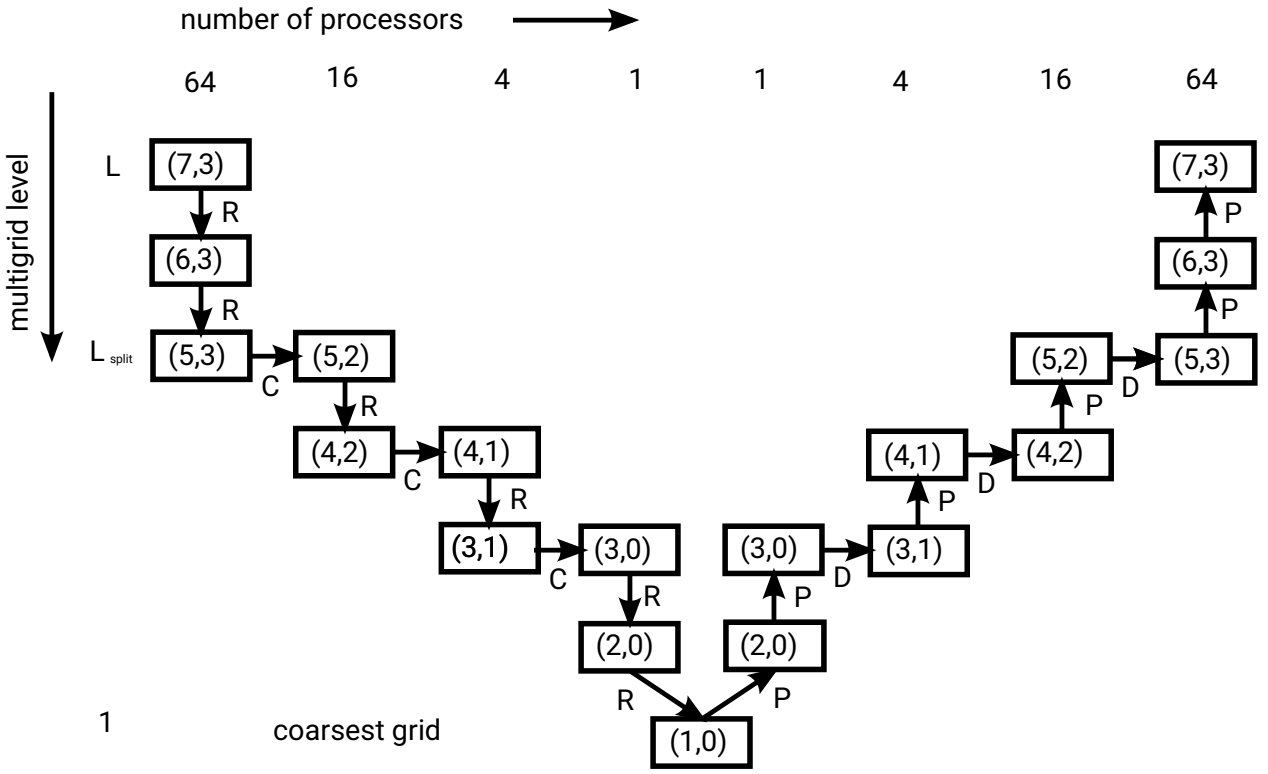
  - Premier portage sur GPU de la version « Stand Alone » fortran90 (CALMIP & IDRIS)
    - Refactoring des tableaux KJI → IJK , pour « coalescence mémoire » sur GPU
    - Rajout d'une dizaine de directives «!$acc kernel »
    - Compilation pour exécution en parallèle, à la fois :
      - En multicore pour le CPU ↔ ACC_NUM_CORE = OMP_NUM_THREADS
      - et « Managed Memory » pour le GPU↔pas besoin des directives «!$acc data »
        « -ta=multicore,tesla:managed »

        => code bit-reproductible entre le CPU // et le GPU

Solveur Multi-grille Géométrique

«TensorProductMultiGrid» :

Collecte/Distribution des grilles / fc(Nb_processeurs)



Méso-NH <=> HPCG
enfin un bench qui sert à quelque chose !

https://www.hpcg-benchmark.org → Results → June 2019

June 2019 HPCG Results

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak
1	DOE/SC/ORNL USA	**Summit** – AC922, IBM POWER9 22C 3.07GHz, dual-rail Mellanox EDR Infiniband, NVIDIA Volta V100 IBM	2,414,592	148.600	1	2.926	1.5%
2	DOE/NNSA/LLNL USA	**Sierra** – S922LC, Power9 22C 3.1GHz, Mellanox EDR, NVIDIA Tesla V100 IBM / NVIDIA / Mellanox	1,572,480	94.640	2	1.796	1.4%
3	Riken Center for Computational Science Japan	**K computer** – , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10.510	20	0.603	5.3%
4	DOE/NNSA/LANL/SNL USA	**Trinity** – Cray XC40, Intel Xeon E5-2698 v3 16C 2.3GHz, Aries, Intel Xeon Phi 7250 68C 1.4GHz Cray	979,072	20.159	7	0.546	1.3%
5	National Institute of Advanced Industrial Science and Technology (AIST) Japan	**AI Bridging Cloud Infrastructure (ABCI)** – PRIMERGY CX2570M4, Intel Xeon Gold 6148 20C 2.4GHz, Infiniband EDR, NVIDIA Tesla V100 Fujitsu	391,680	19.880	8	0.509	1.6%
6	Swiss National Supercomputing Centre (CSCS)	**Piz Daint** – Cray XC50, Intel Xeon E5-2690v3 12C 2.6GHz, Cray Aries, NVIDIA	387,872	21.230	6	0.497	1.8%

Mes
mesoscale no

Debuggage Aller Retour
PC <=> Cluster Local <=> Centre Régional <=> Tier 0