Version GPU de CROCO

Plan

Table of Contents

Matos & Softs

- Centres nationaux, mesocentre équipés de GPUS, nuwa, Pc+carte
- Compilateurs : Nvidia, gcc, gfortran,... $\rightarrow$ gnu=work in progess
- profiles,deburgers : Nvidia; Alinéa DDT; Tau; Vampir; ...
- **MesoNET**

Centre	Machine
TGCC	Partition Joliot-Curie/Irene Rome, Bull (CPU)
	Partition Joliot-Curie/Irene SKL, Bull Skylake (CPU)
	Partition Joliot-Curie/Irene V100, Bull Cascade Lake et V100 (GPU)
	Partition Prototype QLM40, Proto quantique, simulateur 40 Qbits
	Partition Prototype ARM A64FX, Proto Fujitsu ARM A64FX
IDRIS	Partition Jean Zay CSL, HPE Cascade Lake (CPU)
	Partition Jean Zay V100, HPE Cascade Lake (CPU) et V100 (GPU)
	Partition Jean Zay A100, HPE EPYC Milan (CPU) et A100 (GPU)
CINES	Partition Adastra Genoa, HPE EPYC 9654 (CPU 4e gen.)
	Partition Adastra MI250x, HPE EPYC Trento (CPU 3e gen.) et MI250x (GPU)

$\rightarrow$ OpenACC & Nvidia

Table of Contents

OpenACC c'est simple

Kernel

```
1 !$acc kernels
2       do j=Jstr,Jend
3         do i=Istr,Iend
4           zeta(i,j,tndx)=zeta(i,j,
      tndx) - mssh
5         enddo
6       enddo
7 !$acc end kernels
8
```

update et copy

```
1 !$acc enter data copyin(zeta)
2
```

```
1 !$acc kernels default(present)
2       do j=Jstr,Jend
3         do i=Istr,Iend
4           zeta(i,j,tndx)=zeta(i,j,tndx) - mssh
5         enddo
6       enddo
7 !$acc end kernels
8
```

```
1 !$acc data copy(A) create(Anew)
2 while ( (error > tol)  .and.  (iter  <  iter_max ) )
3       error = 0.0
4 !$acc acc kernels
5 !$acc loop independent collapse(2) reduction(max:error)
6       do j=Jstr,Jend
7         do i=Istr,Iend
8           zeta(i,j,tndx)=zeta(i,j,tndx) - mssh
9           error = max( error, zeta(i,j,tndx) )
10        enddo
11      enddo
12 !$acc end kernels
```

https://www.openacc.org/

Table of Contents

OpenACC dans croco

Tout le code est sur GPU

- cppdefs.h : **#define OPENACC**
- main : Envoie des données vers le GPU
  - copy_to_devices.F : common2device.py
- kernels
  - kernels simples
  - Boucles 3D, change_loops.py : DOLOOP2D DOEXTEND
- retour des donnée du GPU vers le CPU
  - xios ou wrt_his
  - diags
  - ...
- Debug

change_loops.py : DOLOOP2D et DOEXTEND

omega.F

```
!       do j=Jstr,Jend
DOLOOP2D(Istr,Iend,Jstr,Jend)
      do i=Istr,Iend
          We(i,j,0)=0. ; FC(i,0)=0. ;
        DC(i,0)=0.
      enddo
      do k=1,N,+1
          do i=Istr,Iend
              FC(i,k)=FC(i,k-1) -...
              DC(i,k)=DC(i,k-1) + Hz(
       i,j,k)
          enddo
      enddo
      do i=Istr,Iend
          DC(i,N)=FC(i,N)/DC(i,N)
      enddo
      do k=1,N-1,+1
          do i=Istr,Iend
              We(i,j,k)=FC(i,k)-DC(i,
       N)*DC(i,k)
          enddo
      enddo
      do i=Istr,Iend
          We(i,j,N)=0.
      enddo
!
! enddo  ! j loop
ENDDOLOOP2D

```

omega_.f

```
!$acc kernels if(compute_on_device )default(
      present)
    DO j=Jstr,Jend
!$acc loop private(DC1D,FC1D) vector
        DO i=Istr,Iend
        We(i,j,0)=0.D0 ; FC1D(0) = 0.D0 ; DC1D(0)
      =0.D0
            do k=1,N,+1
                FC1D(k)=FC1D(k-1) -...
                DC1D(k)=DC1D(k-1) + Hz(i,j,k)
            enddo
            DC1D(N)=FC1D(N)/DC1D(N)
            do k=1,N-1,+1
                We(i,j,k)=FC1D(k)-DC1D(N)*DC1D(k)
            enddo
            We(i,j,N)=0.D0
        ENDDO
    ENDDO

```

compilation

omega_tile: 316, Generating default present(....)
317, Loop is parallelizable
319, Loop is parallelizable
Generating NVIDIA GPU code
317, !$acc loop gang, vector(4)
319, !$acc loop gang, vector(32)
323, !$acc loop seq
319, Local memory used for fc1d,dc1d
323, Loop carried dependence of fc1d prevents parallelization
Loop carried backward dependence of fc1d,dc1d prevents vectorization
Loop carried dependence of dc1d prevents parallelization
Inner sequential loop scheduled on accelerator
Loop carried backward dependence of fc1d,dc1d prevents vectorization
331, Loop is parallelizable

Data → *CPU*

Pas de GPU dans XIOS
send_xios_diags.F

```
1       if (xios_field_is_active("zeta")) then
2 !$acc update if(compute_on_device) host ( zeta )
3       call xios_send_field("zeta",zeta(:,:,fast_indx_out))
4       endif
```

wrt_his.F

```
1       if (wrthis(indxZ)) then
2  !$acc update if(compute_on_device)  host(zeta(:,:,fast_indx_out))
3         work2d=zeta(:,:,fast_indx_out)
4  #if defined WET_DRY && !defined ZETA_DRY_IO
5         do j=0,Mm
6           do i=0,Lm
7             if (h(i,j) .le. Dcrit(i,j)) then
8               work2d(i,j)=work2d(i,j)+h(i,j)
9             endif
10          enddo
11        enddo
12 #endif
```

Table of Contents

Perfs

Strong scaling of the BASIN Hydrostatic.
BASIN 2048x2048 points on the horizontal grid and 50 vertical levels, 800 iterations.
209.71 millions gridpoints
performance : timesteps/ second per milliongridpoints

Nb Noeuds	1	2	4	6	8
tps-CPU	5970.8	2698.3	1164.4	730.2	523.6
idéal-CPU	5970.8	2985.4	1492.7	995.1	746.3
performance	**28.09**	**62.17**	**164.458**	**229.747**	**320.37**
idéal-performance	28.09	56.19	112.38	168.58	224.77
tps-GPU	436.5	268.2	181.1	131.6	98.4
idéal-GPU	436.5	218.2	109.1	72.7	54.57
performance	**384.28**	**625.53**	**926.02**	**1274.44**	**1687.80**
idéal-performance	384.28	768.56	1537.12	2305.68	3074.24
tpsCPU/tpsGPU	13.68	10.06	6.43	5.55	5.32

Perfs

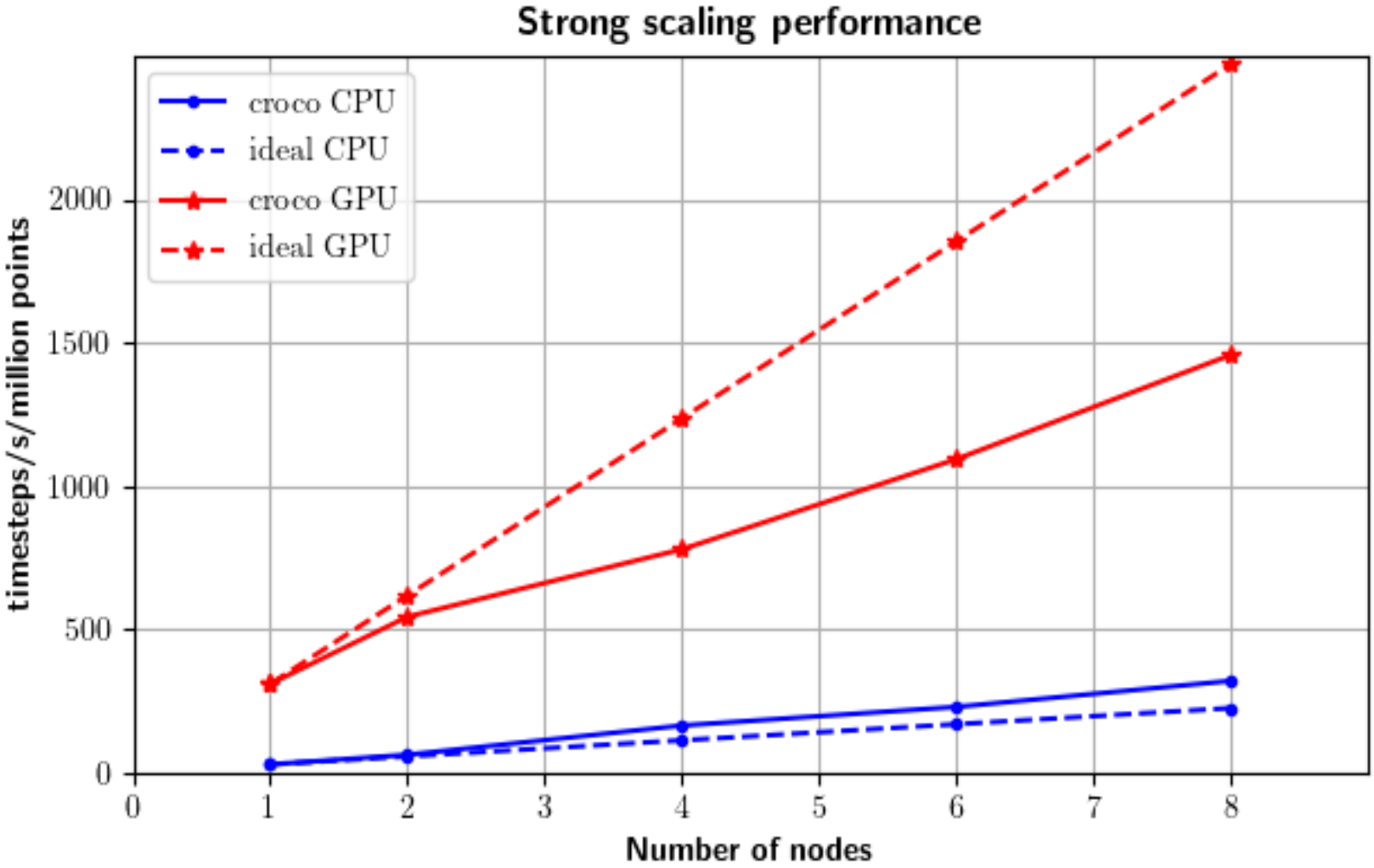
Strong scaling performance
croco CPU
ideal CPU
croco GPU
ideal GPU
timesteps/s/million points
0
500
1000
1500
2000
0
1
2
3
4
5
6
7
8
Number of nodes

Table of Contents

On en est où

- branche dev2022_GPU_merge en phase avec le master
  - CPU Ok
  - GPU en test
  - AGRIF : bientôt
- dev_2022_GPU_NBQ2
  - Cas test OK sur GPU, bonne perf
  - Gibraltar...
- maintenance
  - gitlab-ci ?
  - psyclone

questions ?