Pôle de Calcul Intensif pour la mer , 11 Decembre 2009

"Premiers résultats sur le portage de *NHOES (NonHydrostatic Ocean model for Earth Simulator)* en GPU, en utilisant **HMPP**"

Guillaume PESQUET(ENSIETA)

Tina ODAKA (IFREMER)

ifremer

ENSIETA
ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS

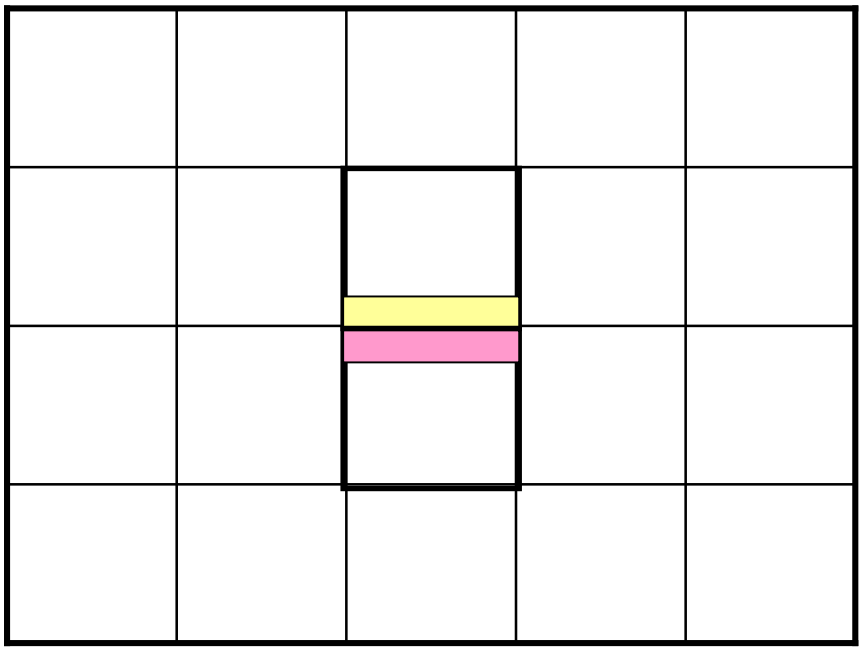
NHOES

- ***NonHydrostatic Ocean model for Earth Simulator***
  - Used by LPO, Bach-Lien HUA's group.
  - IBM **Power6 at** IDRIS:vargas
    - 1600x1600x1600 sur 512 procs
  - NEC at JAMSTEC: Earth simulator
    - 2600x2600x2600 sur 512 procs
- 'One subroutine use most of computational time.'
- Code is simple (only 18 000 lines), comments are well documented, subroutines are well structures.

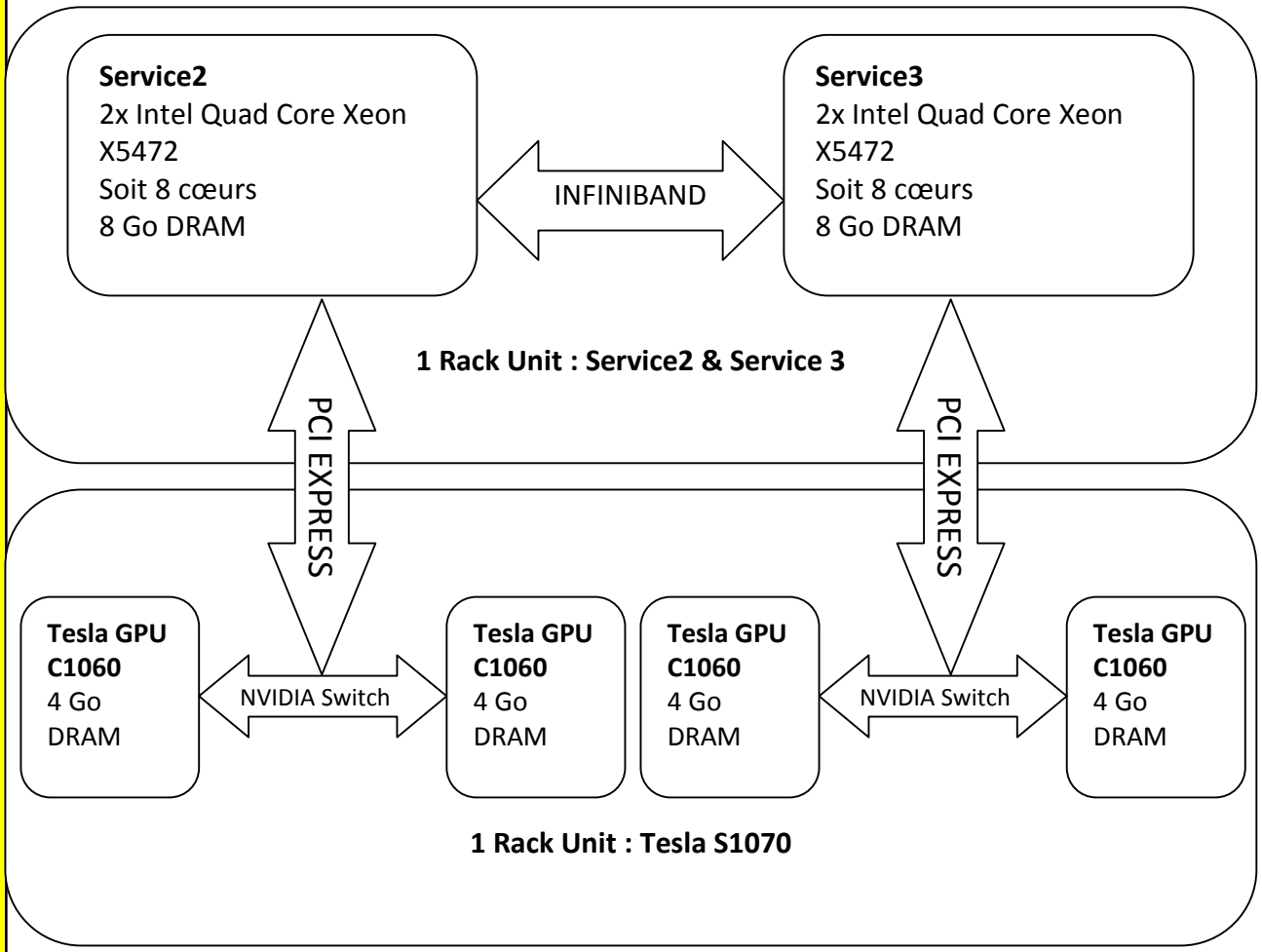
Profiling

% time	cumulative % time	calls	name
23,74	23,74	9	biharmonic_stress_
15,49	39,23	9	momentum_equation_
8,37	47,6	9	absolute_vorticity_
8,05	55,65	9	biharmonic_diffusion_
5,75	61,4	9	kinetic_energy_
5,7	67,1	9	tracer_advection_
5,38	72,48	8	diag_vert_flow_
5,36	77,84	9	barotropic_mode_
4,8	82,64	9	next_tracer_
3,79	86,43	8	next_uvw_
2,05	88,48	9	get_pressure_

NHOES:MPI, excg



Connection between CPU and GPU



PCI express
Is VERY SLOW
compare to
GPU cycle.

One must avoid
overhead by
transfering data
between CPU
and GPU

NHOAS: structure of the code

main

Call …

Do

call one_cycle

enddo

Call …

one_cycle

- Call ….
- Call excg_tracer
- Call Biharmonic_stress
- Call hydrostatic_pressure
- Call absolute_vorticity
- Call kinetic_energy
- Call momentum_equation
- Call excg_mgradient
- Call ….

Biharmonic_strees

```
do k = KL, KM0
   do j = JL, JM0
      do i = IL, IM0
                  ….
      end do
   end do
end do
…
```

Call excg_uvw

```
do k = KL, KM0
   do j = JL, JM0
      do i = IL, IM0
                  ….
      end do
   end do
end do
…
```

NH2: structure of the code

main

Call …

Do

call one_cycle

enddo

Call …

one_cycle

- Call ….
- Call excg_tracer
- Call gpu1
- Call excg_uvw
- Call gpu2
- Call excg_mgradient
- Call ….

gpu1

(Biharmonic_strees1)

```
do k = KL, KM0
   do j = JL, JM0
      do i = IL, IM0
                 ....
      end do
   end do
end do
...
```

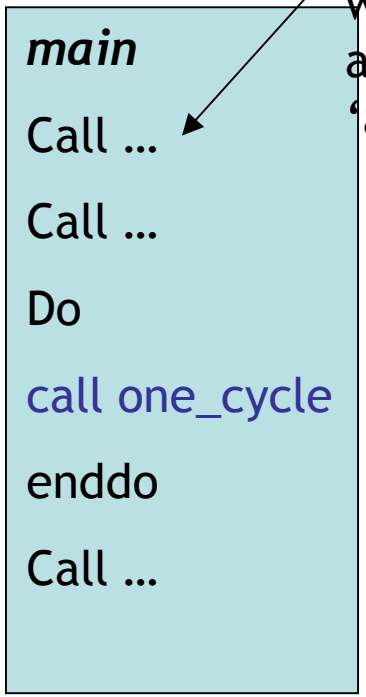
Gpu2

(Biharmonic_strees2)

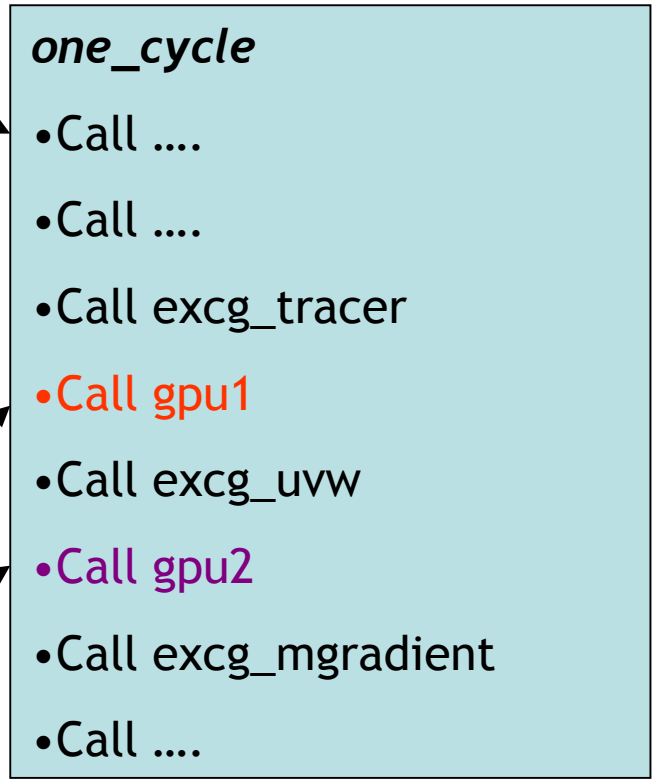
```
do k = KL, KM0
   do j = JL, JM0
      do i = IL, IM0
                 ....
      end do
   end do
end do
...
```

- hydrostatic_pressure
- absolute_vorticity
- kinetic_energy
- momentum_equation

NH2 => NH(GPU)
Optimise data transfer



Send datas needed in ‘gpu1’ and ‘gpu2’ when those datas are calculated ‘advance load’

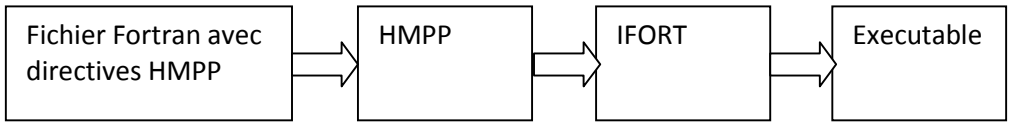


Keep as much data
As possible on GPU
Between gpu1 and gpu2
=>
Bring only the data which are exchanged between nodes from cpu.

Compiling with HMPP

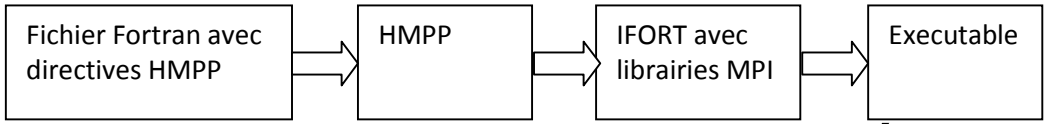
- HMPP

  hmppfort --compiler ifort input.F90

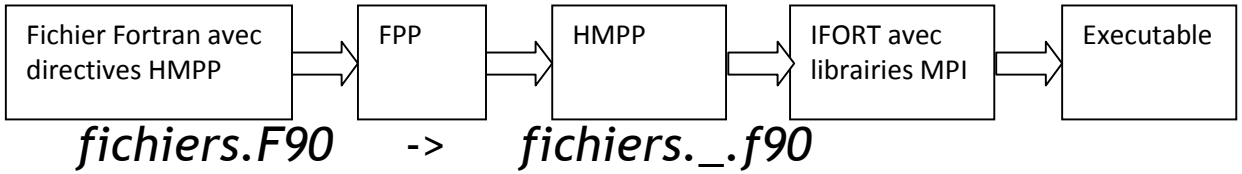


- MPI

  hmppfort --compiler ifort -L/appli/intel/impi/3.2.1.009/lib64 -lmpiif -L/appli/intel/impi/3.2.1.009/lib64 -lmpi input.F90 -o output –force

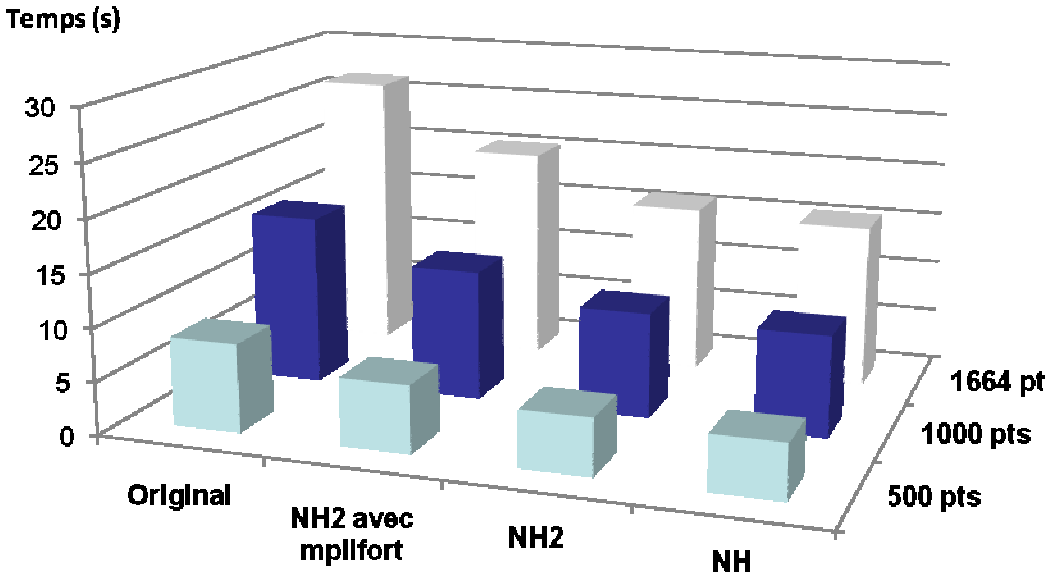


- MPI avec instructions préprocesseur



Result

Kmax	500	1000	1664
Imax jmax	100 x100	100 x100	100 x100
Original Cpu only	8,3 s	16,1 s	26,5 s
NH2 cpu only	6,3 s	12,3 s	20,3 s
NH2 Gpu + cpu	5,3 s	9,9 s	16 s
NH Gpu + cpu	**5,1 s**	**9,5 s**	**15,5 s**
Ratio	1.62	1.69	1.71
Ratio	1,24	1,29	1, 31



Ca 50 % of cpu time is now sent to GPU.
i.e. pure GPU speed up for ‘gpu1’ and ‘gpu2’ is about ‘2’.

Kmax 1664 comes from tesla have 4G of memory. Imax jmax can extend by using more mpi procs.

What's next?

- MPI->OpenMP
  ->GPU
- Optimise gpu code (codelet tuning)
  - Hmppcg directive
    - Unroll,jam,grid blocksize,shared memory
- Fermi
  - Double precision speed up.
    - Double precision computation on tesla lose 88% of computing resouce.

Questions diverses