Jumping barriers in AROME

Ryad El Khatib
GMAP

Joint 29th ALADIN Workshop & HIRLAM All Staff Meeting
1-5 April 2019
Madrid

Plan

1. The barrier of bounds violation at runtime
2. The barrier of memory bandwidth in the software
3. The first barrier of the extension zone for MPI scalability
4. The second barrier of the extension zone for MPI scalability
5. The barrier of meteorology (!) in the geometry distribution
6. Conclusion

1. The barrier of bounds violation at runtime

Models fail at runtime when the code is compiled with arrays bounds checking option because of several « assumed » violations

- Certain subroutines have to be compiled without bounds checking option
- Some others may be compiled with bounds checking option but the default « undefined » field pointer (NUNDEFLD) should be set to 1

=> Bounds violations cannot be checked properly

Examples : case of CPG_GP

```
REAL(KIND=JPRB), INTENT(IN) :: PGMVT1(NPROMA,NFLEVG,NDIM)

CALL GPINISLB(…, PGMVT1(1,1,MNHX),...)
```

- will fail whenever MNHX < 1 or MNHX > NDIM

Workaround : MNHX=NUNDEFLD=1

=> Consequence : any true bounds violation is masked by NUNDEFLD=1

Suggestion :
use a generic function addressing a pointer

```
USE SC2PRG_MOD       , ONLY : SC2PRG

REAL(KIND=JPRB), INTENT(IN) :: PEXTRA(NPROMA,NVEXTRDYN,NGPBLKS)
TYPE(TRAJ_TYPE_OOPS),OPTIONAL,INTENT(INOUT)    :: PTRAJEC_OOPS
REAL(KIND=JPRB), INTENT(IN) :: PGMVT1(NPROMA,NFLEVG,NDIM)

REAL(KIND=JPRB), POINTER :: ZEXTRA(:,:,:)
TYPE(TRAJ_PHYS_TYPE), POINTER :: ZTRAJEC_PHYS
REAL(KIND=JPRB), POINTER :: ZNHXT1(:,:)

DO JBL=1,NGPBLKS
      CALL SC2PRG(JBL,PEXTRA,ZEXTRA)
      CALL SC2PRG(JBL,PTRAJEC%PHYS,ZTRAJEC_PHYS)
      CALL CPG(…, ZEXTRA, …,ZTRAJEC_PHYS)
ENDDO

CALL SC2PRG(MNHX,PGMVT1,ZNHXT1)
CALL GPINISLB(…, ZNHXT1,...)
```

SIGSEGV at runtime in case of true bounds violation

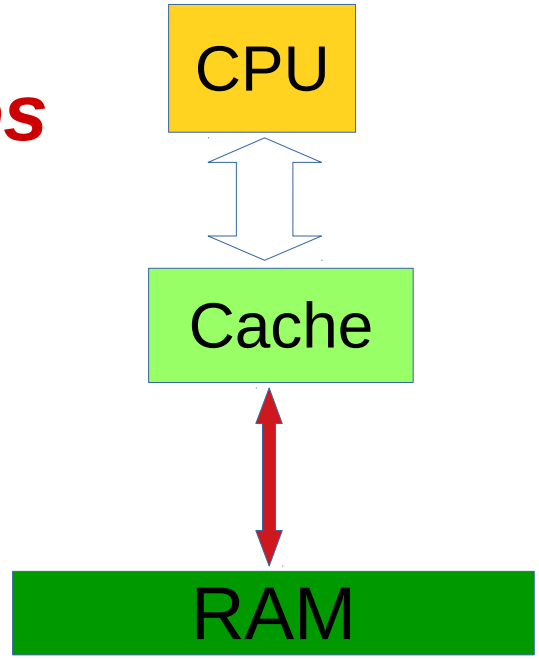
Pros/Cons

- Tested successfully on a few subroutines
- Positive response from ECMWF
- No impact on the performance
- Can be implemented progressively :
    - per subroutine
        - per array
            - per field in an array
- Should make the code easier to read/handle
- ✖ Many lines added/modified
  => automatic transformation with a script is needed
  (and on track (- may be tested/adopted with cycle 47 ?)

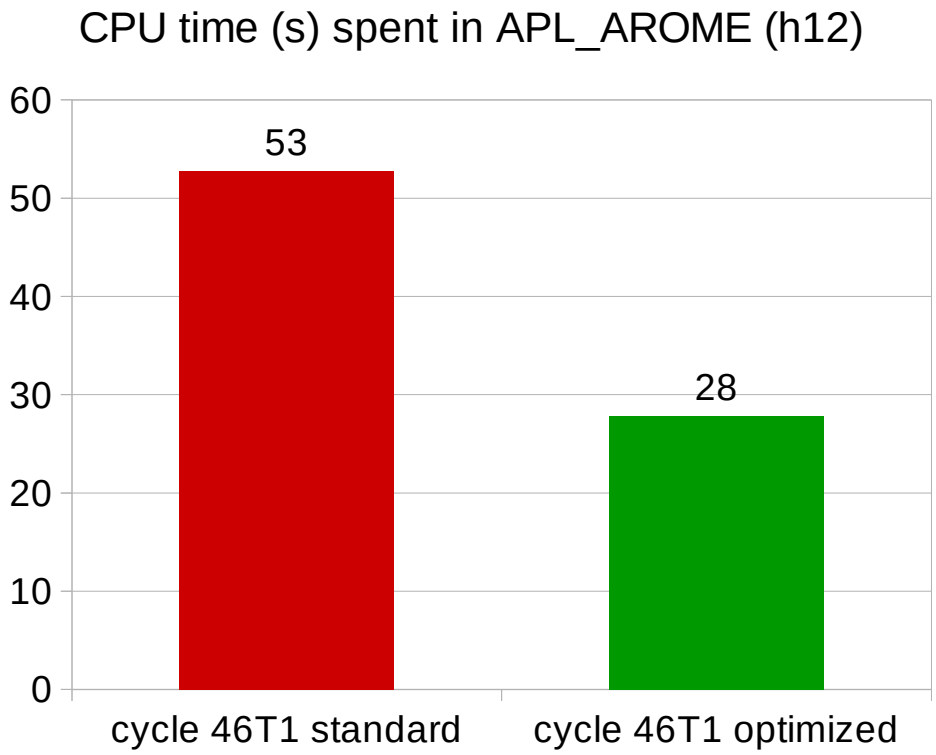
2. The barrier of memory bandwidth

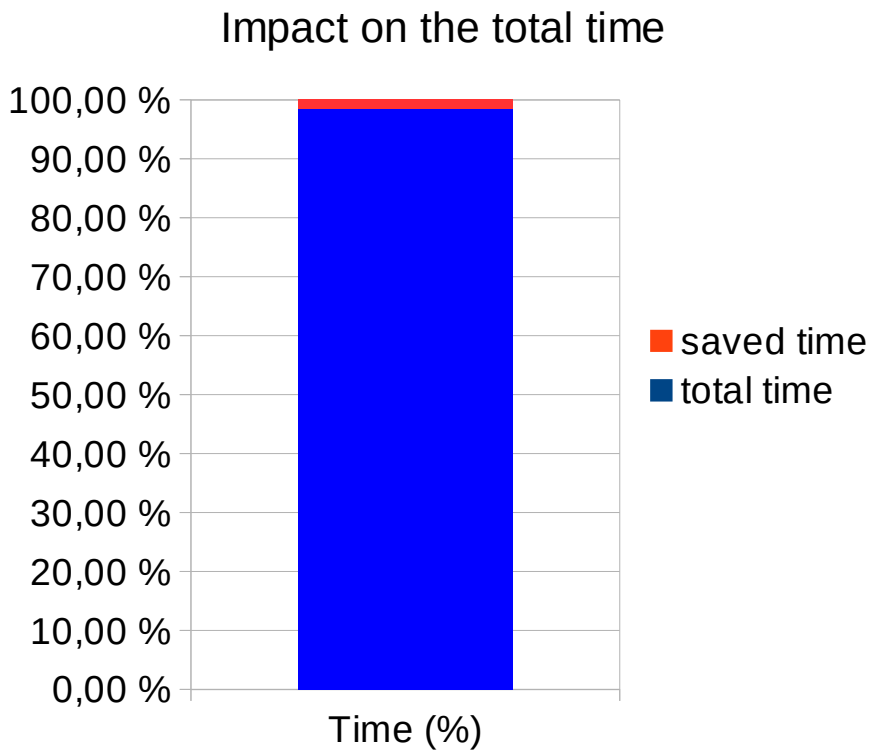
AROME is sensitive
- not the *amount of memory* used
- but the amount of ***memory accesses***

Caused by :
- arrays initialisations
- arrays copies
- array syntax in calculation
  (less cache re-use)



Memory bandwidth : the impact of APL_AROME





1.6 % total time saved … just by removing useless initializations or copies of array

How to remove safely useless initializations of array

```
ZX(:)=0.    <= Useless
ZY(:)=0.    <= Needed

DO J=1,N
  ZX(J)=F(J)
  ZY(J)=ZY(J)+ZX(J)
ENDDO
```

<= Old code

Proposed code =>

Conditional initialization Allowing debugging **=>**

```
INIT0=-1
IF (INIT0 == 0) THEN
  ZVALUE=HUGE(1.)
ELSE
  ZVALUE=0.
ENDIF

 IF (INIT0 >= 0) THEN
   ZX(:)=ZVALUE
   ZY(:)=ZVALUE
 ENDIF

 DO J=1,N
   ZX(J)=F(J)
   ZY(J)=0.
   ZY(J)=ZY(J)+ZX(J)
 ENDDO
```

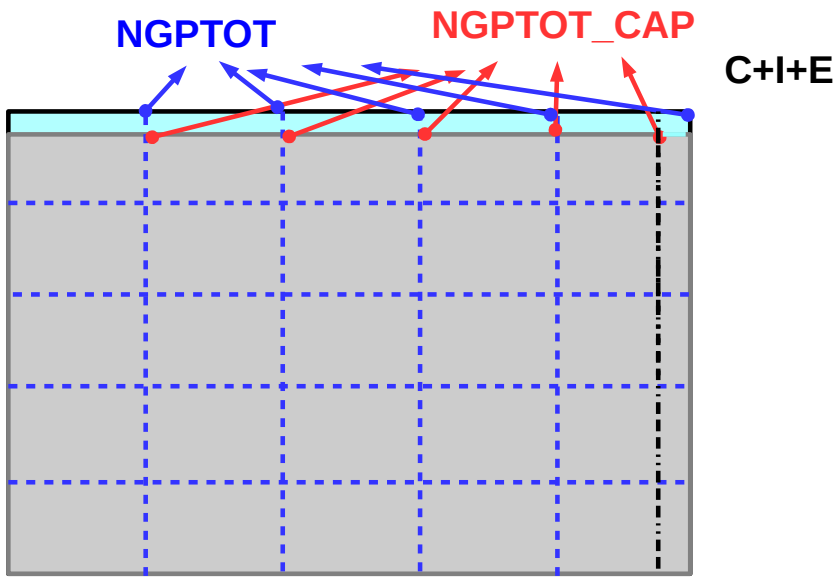
INIT0 = 0 : initialization to HUGE
INIT0 = 1 : initialization to a realistic value
INIT0 =-1 : No initialization at all

Necessary initialisation close to calculation => => in the memory cache

Many of such opportunities in the code !!

The first barrier of the extension zone for MPI scalability

Northern tasks are imbalanced, compared to the others because gridpoint computation stops at NGPTOT_CAP instead of NGPTOT



Area of gridpoint computation

But should we really stop the computation at NGPTOT_CAP ?

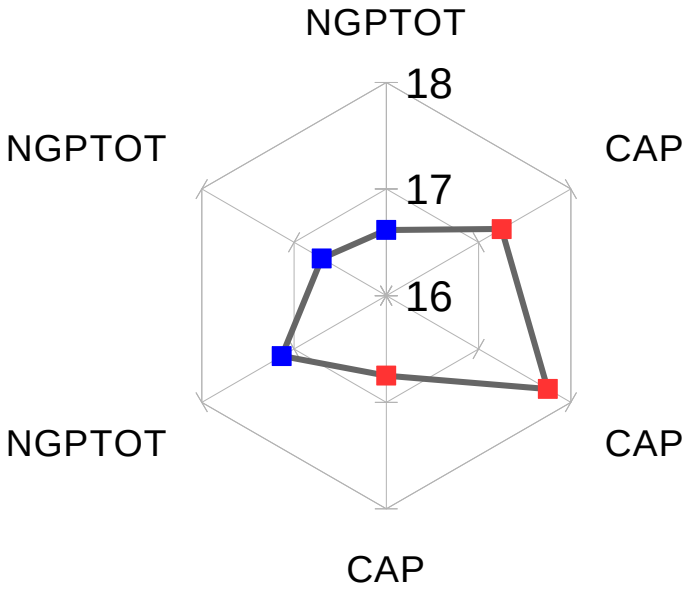
The first barrier of the extension zone for MPI scalability

New namelist key **LGPTOT_CAP** : IF (.NOT. LGPTOT_CAP) NGPTOT_CAP=NGPTOT

real time in minutes of AROME with 1872 MPI tasks, h12

3 runs with NGPTOT, 3 runs with NGPTOT_CAP

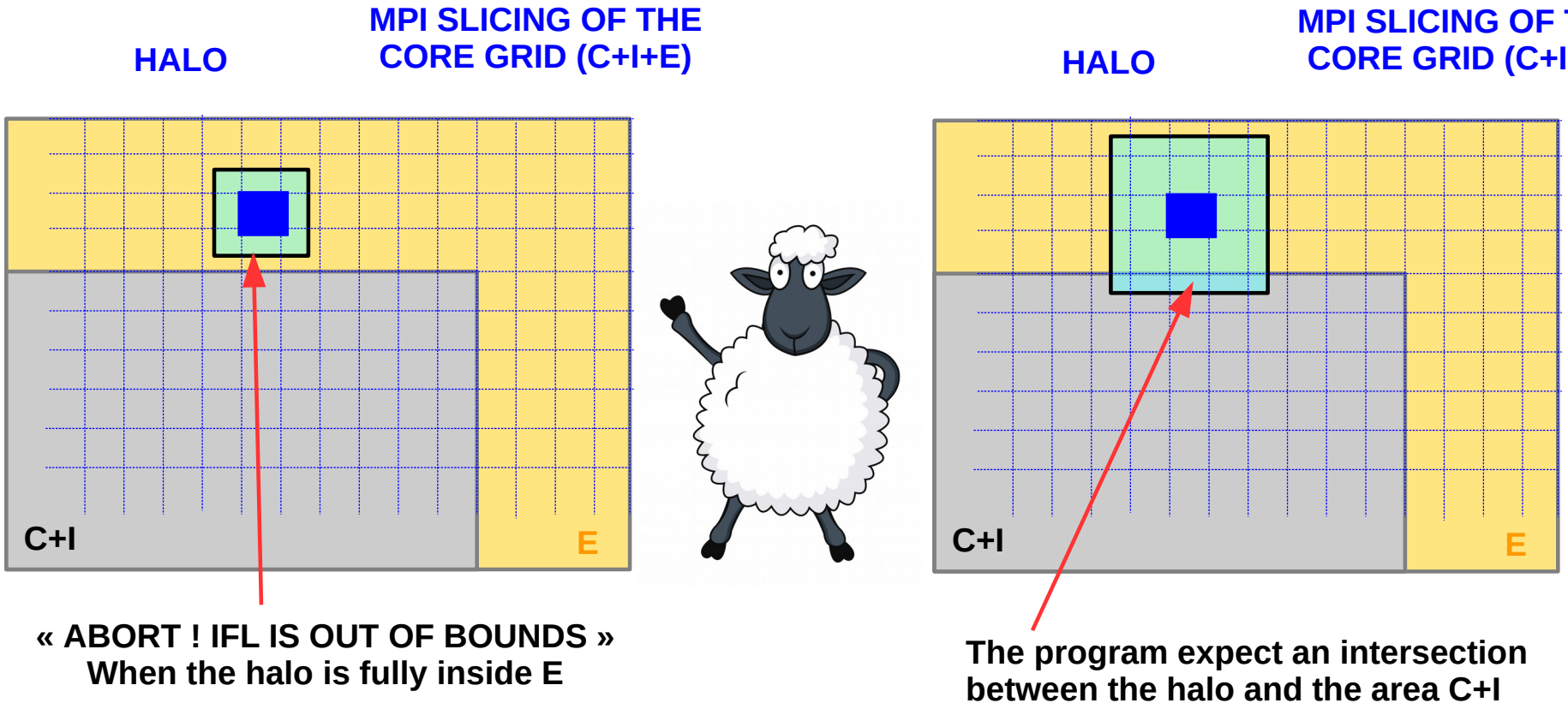
Rather in favour of NGPTOT than NGPTOT_CAP



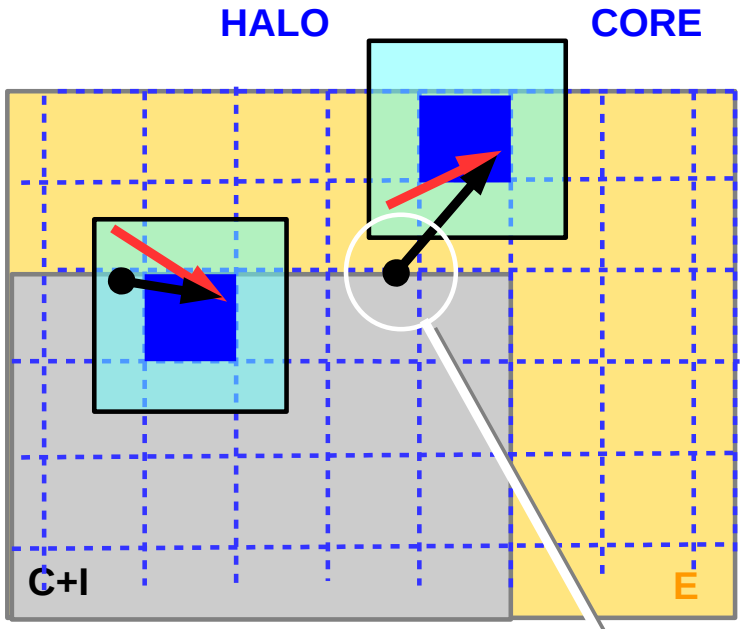
But more tests with more tasks are needed

The second barrier of the extension zone for MPI scalability

ABORT ! SLRSET: IFL IS OUT OF BOUNDS



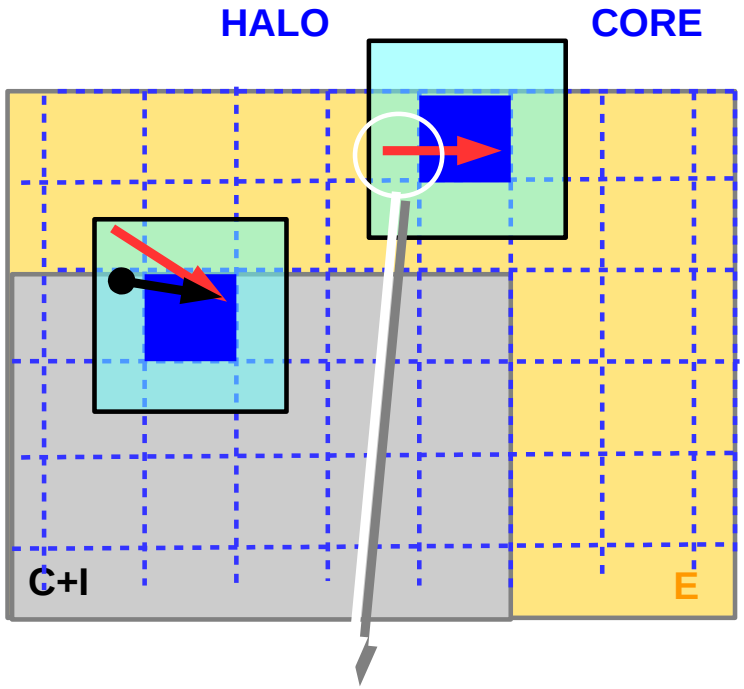
The second barrier of the extension zone for MPI scalability



Raw trajectory from origin point

Corrected trajectory to remain in C+I

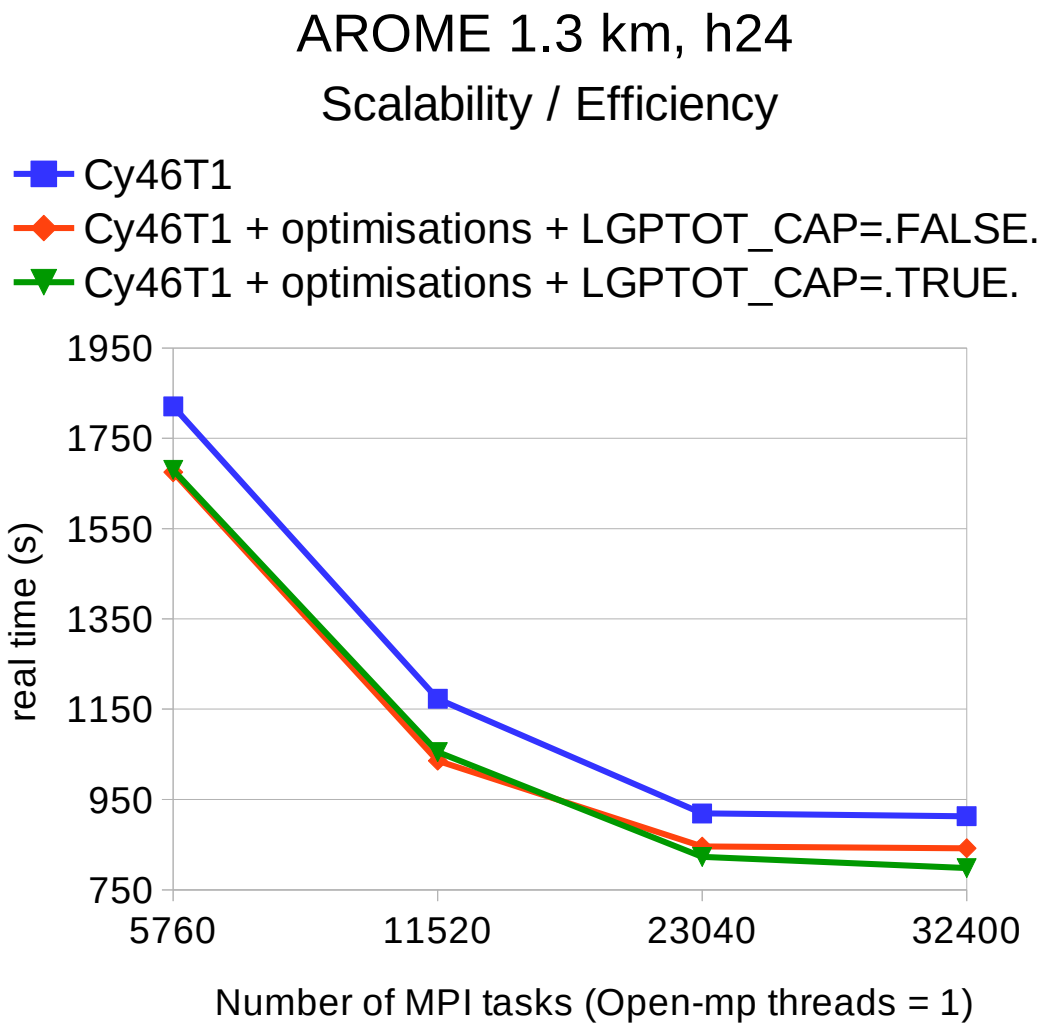
Origin point is out of the halo



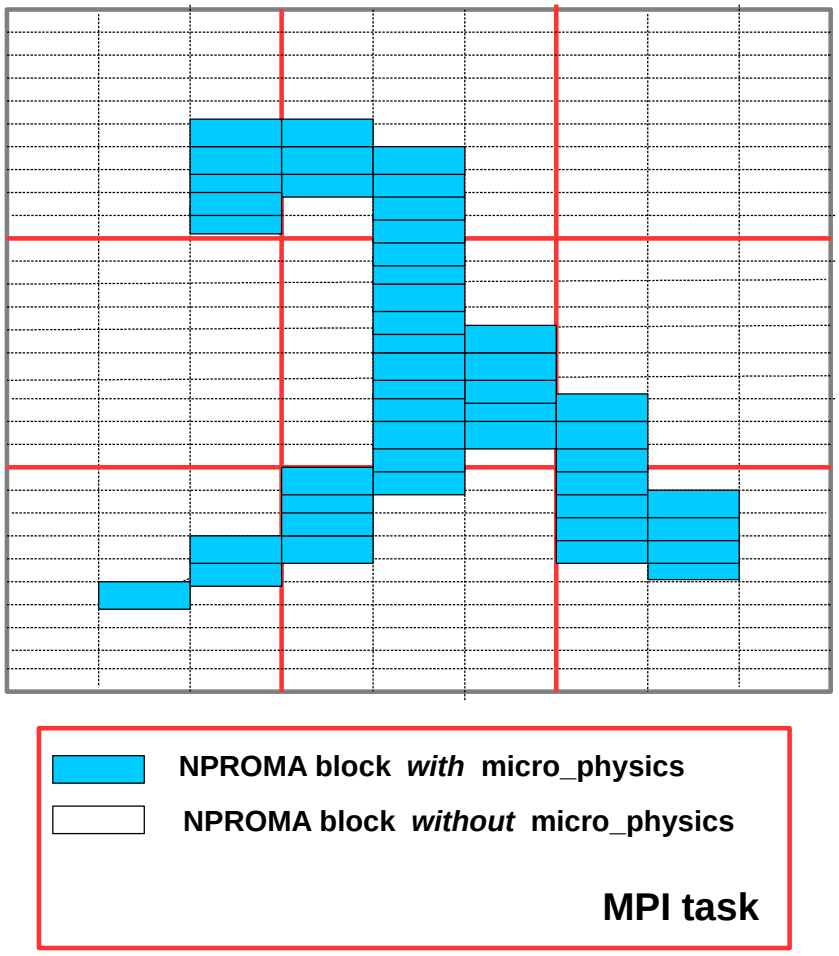
SOLUTION :

- Force points in E-zone to remain along X
- Large scale values will overwrite, anyway

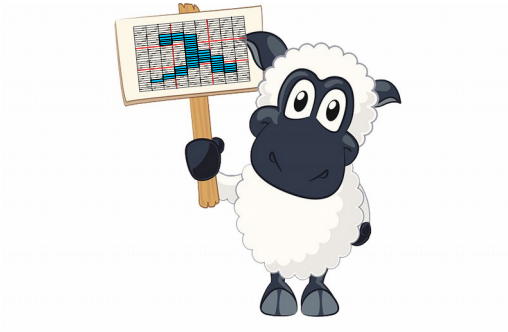
Impact of the optimizations on the scalability and efficiency



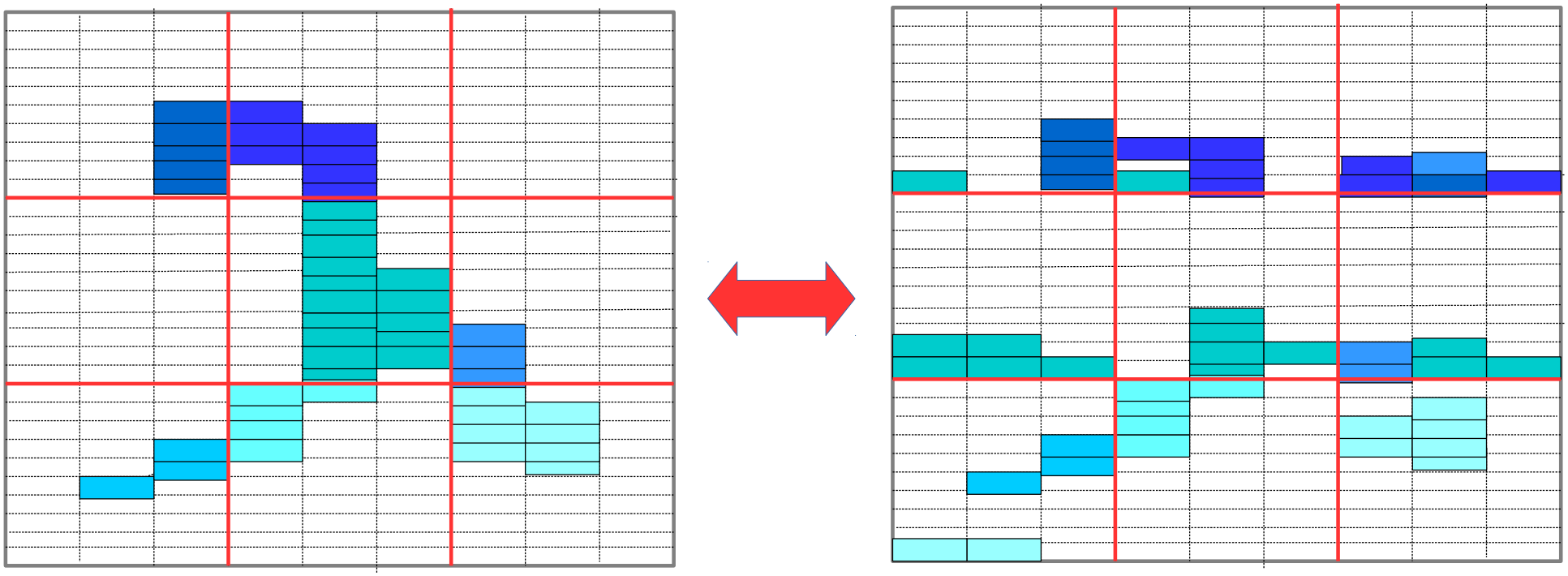
The barrier of meteorology in geometry distribution



Some load imbalance is caused by physical parameterizations



The barrier of meteorology in geometry distribution



Shuffle and re-distribute NPROMA blocs
with non-blocking message passing,
compute,
reshuffle back ..., an utopia or not ?

Conclusion

- Solution to eliminate «assumed» bounds violation on track
- There is room for optimization by reducing the memory bandwidth dependency
- The use of NGPTOT_CAP to limit gridpoint computation in E-zone is still interesting
- No « semi-lagrangian » limit anymore in the number of MPI tasks
- Feasability of a load balancing of physics by shuffling NPROMA blocks : to be studied.

Thank you for your attention !

Météo-France

ryad.elkhatib@meteo.fr

www.meteofrance |@meteofrance

Generic function addressing a pointer

```
MODULE SC2PRG_MOD ! Named in memory of glorious subroutines sc2rdg and sc2wrg

REAL(KIND=JPRB), POINTER :: FAKE2(:,:) => NULL()

INTERFACE SC2PRG
MODULE PROCEDURE SC2PRG2A, ...
END INTERFACE

CONTAINS
SUBROUTINE SC2PRG2A(KBL,PG2A,PGPR)

INTEGER(KIND=JPIM),      INTENT(IN)  :: KBL
REAL(KIND=JPRB),TARGET,  INTENT(IN)  :: PG2A(:,:,:)
REAL(KIND=JPRB),POINTER :: PGPR(:,:)

IF (SIZE(PG2A) == 0) THEN
  PGPR => FAKE2
ELSEIF ( (LBOUND(PG2A,DIM=3) <= KBL) .AND. (KBL <= UBOUND(PG2A,DIM=3)) ) THEN
  PGPR => PG2A(:,:,KBL)
ELSE
  PGPR => FAKE2
ENDIF

END SUBROUTINE SC2PRG2A

END MODULE SC2PRG_MOD
```