ACCORD
A Consortium for COnvection-scale modelling Research and Development

Guidelines for refactoring the physics in view of GPU porting

Daan Degrauwe

ALARO-1 Working Days

13-15 June 2022, Prague

Background

- ECMWF: internal Hybrid24 project
- MeteoFrance: next procurement (end 2023)
- DestinE/DEODE: targeting ~200m resolution, relies on (GPU-powered) EuroHPC infrastructure
- ACCORD:
  - maintain performance on existing machines!
  - Affect scientists as little as possible
  - Keep single code base

Aim of refactoring

Flexibility is key, esp. in the physics parameterizations

- In terms of granularity of parallel loops:
  - Support one big top-level OpenMP loop (current layout)
  - Support many smaller lower-level OpenMP/OpenACC loops
- In terms of target architecture

  ⇒ Use “smart” (i.e. accelerator-aware) data structures to avoid modifying entire code when targeting new hardware

  ⇒ Use source-to-source transformation tools to adapt code to specific hardware:
    - Loop reordering
    - Annotation with accelerator directives
    - Array promotion/demotion

Status

Both types of flexibility are realized for ARPEGE physics by working on 3 fronts:

- Dataflow in top-level routines (CPG_DRV, CPG, MF_PHYS)
- Cleaning of MF_PHYS, extraction of APL_ARPEGE from APLPAR
- Mechanism to choose parallelization granularity at runtime

Status: smart data structures

Introduced at top-level (CPG_DRV/CPG)

Current code:

```
SUBROUTINE CPG_DRV

REAL :: PRE0F(NPROMA,NFLEVG,NBLK)

!$OMP PARALLEL DO
DO JKGLO = 1, KGPCOMP, YDGEOMETRY%YRDIM%NPROMA
  IBL=(JKGLO-1)/YDGEOMETRY%YRDIM%NPROMA+1
  CALL CPG(... , PRE0F(:,:,IBL), ... )
ENDDO
!$OMP END PARALLEL DO

END SUBROUTINE CPG_DRV
```

Status: smart data structures

Introduced at top-level (CPG_DRV/CPG)

New code:

```
MODULE CPG_TYPE_MOD

TYPE CPG_DYN_TYPE

  REAL (KIND=JPRB), POINTER, CONTIGUOUS :: PREF (:, :) => NULL ()
  TYPE (FIELD_3D), POINTER :: F_PREF => NULL ()

  ! ... other fields ...

END TYPE CPG_DYN_TYPE

END MODULE CPG_TYPE_MOD
```

Status: smart data structures

Introduced at top-level (CPG_DRV/CPG)

New code:

```
SUBROUTINE CPG_DRV

TYPE(CPG_PHY_TYPE):: YLCPG_DYN
LOGICAL :: LLPERSISTENT

CALL YLCPG_DYN%INIT (YDFIELDS%REGISTRY, YDCPG_OPTS=YLCPG_OPTS, PERSISTENT=LLPERSISTENT)

!$OMP PARALLEL DO FIRSTPRIVATE(YLCPG_DYN)
DO JKGLO = 1, KGPCOMP, YDGEOMETRY%YRDIM%NPROMA
  IBL=(JKGLO-1)/YDGEOMETRY%YRDIM%NPROMA+1
  CALL YLCPG_DYN%UPDATE_VIEW (BLOCK_INDEX=IBL)
  CALL CPG(YDGEOMETRY, ... , YLCPG_DYN, ... )
ENDDO
!$OMP END PARALLEL DO

END SUBROUTINE CPG_DRV
```

Status: cleaning of MF_PHYS

- Use new data structures; also allows to get rid of LTWOTL
- Move calculations/subroutine calls to APL_*

```
SUBROUTINE MF_PHYS

CALL MF_PHYS_PREP ( ... )
CALL MF_PHYS_INIT ( ... )
CALL YLMF_PHYS_BASE_STATE%INIT ( ... )
CALL YLMF_PHYS_NEXT_STATE%INIT ( ... )

IF (YDMODEL%YRML_PHY_MF%YRSIMPHL%LSIMPH) THEN
  CALL APLSIM( ... )
ELSEIF (YDMODEL%YRML_PHY_MF%YRARPHY%LMPA) THEN
  CALL APL_AROME( ... )
ELSEIF (YDMODEL%YRML_PHY_MF%YRARPHY%LAPL_ARPEGE) THEN
  CALL APL_ARPEGE ( ... )
ELSE
  CALL APLPAR ( ... )
ENDIF

END SUBROUTINE MF_PHYS
```

Status: extraction of APL_ARPEGE from APLPAR

- Partially manually, partially scripted, based on set of logical switches
- (Still contains calculations)
- Local arrays NOT encapsulated or transformed into FIELD_API structures
- Some computations/ subroutine calls grouped into new subroutines

```
SUBROUTINE APL_ARPEGE
[...]
CALL APL_ARPEGE_OCEANIC_FLUXES ( ... )
CALL APL_WIND_GUST ( ... )
CALL APL_ARPEGE_SHALLOW_CONVECTION_AND_TURBULENCE ( ... )
CALL APL_ARPEGE_ALBEDO_COMPUTATION ( ... )
CALL APL_ARPEGE_AEROSOLS_FOR_RADIATION ( ... )
CALL APL_ARPEGE_CLOUDINESS ( ... )
CALL APL_ARPEGE_RADIATION ( ... )
CALL APL_ARPEGE_SOIL_HYDRO ( ... )
CALL APL_ARPEGE_SURFACE ( ... )
[...]
END SUBROUTINE APL_ARPEGE
```

Status: parallelization granularity

- CPG is called with configuration string CDPART

```
SUBROUTINE CPG( ..., CDPART)
IF ( CDPART(1:1) == 'X' ) THEN
  CALL CPG_GP ( ... )
ENDIF
IF ( CDPART(2:2) == 'X' ) THEN
  CALL MF_PHYS ( ... )
ENDIF
IF ( CDPART(3:3) == 'X' ) THEN
  CALL CPG_DIA ( ... )
  CALL CPG_DYN ( ... )
  CALL CPG_END ( ... )
ENDIF
END SUBROUTINE CPG
```

- This allows to decide at runtime to keep coarse granularity (‘XXX’) or to split in smaller parts (‘X00’,’0X0’,’00X’)

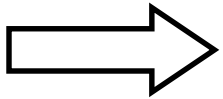
Status: parallelization granularity

- Finer granularity at lower levels (MF_PHYS, APL_ARPEGE, below) is achieved by preprocessing the code and generating *_PARALLEL.F90 files

```
SUBROUTINE MF_PHYS( ... )

!=PARALLEL
CALL APL_ARPEGE( ... )
!=END PARALLEL

END SUBROUTINE MF_PHYS
```



```
SUBROUTINE MF_PHYS_PARALLEL( ... )


CALL APL_ARPEGE_PARALLEL( ... )


END SUBROUTINE MF_PHYS_PARALLEL
```

Status: parallelization granularity

Other modifications done by preprocessor (Perl scripts based on fxtran Fortran parser):

- replace subroutine calls with calls to processed routines (identified by adding _PARALLEL to their name).
- replace raw Fortran arrays with FIELD API objects, set pointers to their data
- add a loop over the NPROMA blocks;
- add OpenMP directives around this loop;
- add the NPROMA block enumerator to arrays in calculations and subroutine calls

Status: parallelization granularity

Example:

```
SUBROUTINE APL_ARPEGE( ... )

REAL(KIND=JPRB) :: &
 & ZDCAPE(YDCPG_OPTS%KLON)

!=PARALLEL
CALL ACEVADCAPE(ZDCAPE)
DO JLEV = 1, KFLEVG
  DO JLON = KIDIA, KFDIA
    ZQGM(JLON,JLEV)=ZEPSNEB
  ENDDO
ENDDO
!=END PARALLEL

CALL APL_ARPEGE_RADIATION(...)

END SUBROUTINE MF_PHYS
```

```
SUBROUTINE APL_ARPEGE_PARALLEL

TYPE (FIELD_2D), POINTER :: YL_ZDCAPE
REAL(KIND=JPRB), POINTER :: ZDCAPE(:,:)

ZDCAPE => GET_HOST_DATA_RDWR (YL_ZDCAPE)
Z_YDMF_PHYS_OUT_UCLS => &
 & GET_HOST_DATA_RDONLY (YDMF_PHYS%OUT%F_UCLS)

!$OMP PARALLEL DO PRIVATE (JBLK, JLEV, JLON, YLCPG_BNDS)
DO JBLK = 1, YDCPG_OPTS%KGPBLKS
  YLCPG_BNDS = YDCPG_BNDS
  CALL YLCPG_BNDS%UPDATE (JBLK)
  CALL ACEVADCAPE( ZDCAPE(:,JBLK) )
  DO JLEV = 1, YDCPG_OPTS%KFLEVG
    DO JLON = YLCPG_BNDS%KIDIA, YLCPG_BNDS%KFDIA
      ZQGM(JLON,JLEV,JBLK)=ZEPSNEB
    ENDDO
  ENDDO
ENDDO

CALL APL_ARPEGE_RADIATION_PARALLEL(...)

END SUBROUTINE APL_ARPEGE_PARALLEL
```

ACCORD
A Consortium for COnvection-scale modelling Research and Development

Status: parallelization granularity

Notes on preprocessor approach:

- Flexibility in terms of parallelization granularity: fine granularity now amounts to 80 separate OpenMP loops for APL_ARPEGE, but this can be changed by merging/splitting “!=PARALLEL” regions
- Preprocessing is only necessary when targeting non-CPU machine
- Code changes on development code are rather limited*:
  - No FIELD_API objects, just raw Fortran arrays
  - Adding !=PARALLEL directives
- No overhead (memory/walltime) due to FIELD_API objects

* but not inexistent; see guidelines later

Performance

3 experiments with ARPEGE on TL1798L105 grid

	Reference	Persistent	Parallel
CPG	DO JBLK=1,NBLK CALL CPG_GP CALL MF_PHYS CALL CPG_DIA ! … ENDDO	DO JBLK=1,NBLK CALL CPG_GP ENDDO DO JBLK=1,NBLK CALL MF_PHYS ENDDO DO JBLK=1,NBLK CALL CPG_DIA ! … ENDDO	DO JBLK=1,NBLK CALL CPG_GP ENDDO CALL MF_PHYS_PARALLEL DO JBLK=1,NBLK CALL CPG_DIA ! … ENDDO
MF_PHYS/ MF_PHYS_PARALLEL	CALL APL_ARPEGE	CALL APL_ARPEGE	CALL APL_ARPEGE_PARALLEL
APL_ARPEGE/ APL_ARPEGE_PARALLEL	!=PARALLEL CALL ACEVADCAPE !=END PARALLEL CALL APL_ARPEGE_RADIATION	!=PARALLEL CALL ACEVADCAPE !=END PARALLEL CALL APL_ARPEGE_RADIATION	DO JBLK=1,NBLK CALL ACEVADCAPE ENDDO CALL APL_ARPEGE_RADIATION_PARALLEL

ACCORD
A Consortium for COnvection-scale modelling Research and Development

Performance

3 experiments with ARPEGE on TL1798L105 grid

Experiment	RSS memory	Walltime (s)	Walltime w/o setup (s)
Reference	1725	169	133
Persistent	2240	162	144
Parallel	2641	185	165

The conclusion here is NOT that refactoring decreases performance!

The 3 experiments were carried out with the same executable, i.e. increased walltime/memory is only felt by fine-grained parallelization configuration!

Guidelines

ARPEGE physics refactoring exercise resulted in the following guidelines*

- Avoidance of all module variables.
- Avoidance of ALLOCATABLE arrays inside the physics
- All output arrays and local arrays should have dimension NPROMA. Allocation and initialization of arrays that do not satisfy this constraint (e.g. ZVETAH, which characterizes the vertical coordinate) should be moved to the setup, instead of being done inside the gridpoint calculations.
- Constant YDMODEL and YDGEOMETRY
- Use of homogeneous notations throughout the code (esp. loop variables and loop bounds)
- Single routine per file (both for routines inside modules and for routines inside other routines). A possible workaround is a preprocessing script which inlines the contained subroutine.

* Not all of these are to be taken too strictly

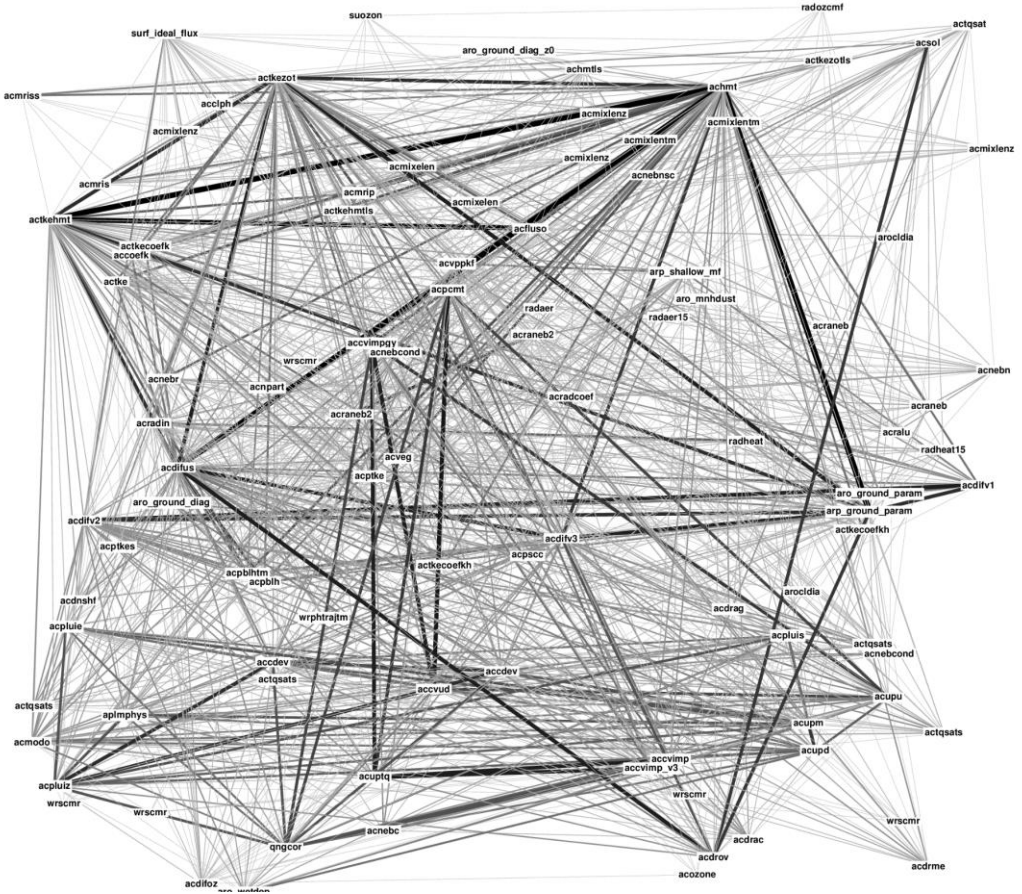
Timeline

- All above phased into cy48t3, to be declared by end-June
- cy49 build:
  - start immediately after declaration of cy48t3
  - declaration by end-September
- cy49t1
  - contribution window: October 2022 – January 2023
  - (AROME refactoring status by Jan 2023 not clear)
- For ALARO:
  - Refactoring/cleaning is not urgent per se, can be done progressively
  - Keep ISBA in refactored code?
  - What about other ongoing scientific ALARO developments?
  - Organize working week still in 2022

Conclusions

- Impressive progress on ARPEGE refactoring
- Satisfying important constraints:
  - No negative impact on performance on CPU machines
  - Limited modifications in low-level scientific code
  - Flexibility of parallelization granularity
- Relying on code preprocessor to adapt code for fine-grained parallelism
- Setting track for LAM configurations

APLPAR


suozon
radozcmf
surf_ideal_flux
actqsat
aro_ground_diag_z0
acsol
actkezotls
achmtls
actkezot
acmriss
achmt
acclph
acmixlenz
acmixlentm
acmixlenz
acmixlentm
acmixlenz
acmixlenz
acmixlen
acnebnsc
acmris
acmrip
acmixelen
actkehmtls
actkehmt
acfluso
arocldia
actkecoefk
accoefk
acvppkf
arp_shallow_mf
actke
acpcmt
aro_mnhdust
radaer
radaer15
acraneb
acraneb2
accvimpgy
acnebcond
wrscmr
acnebn
acnebr
acnpart
acradcoef
acraneb
acradin
acraneb2
acralu
radheat
radheat15
acveg
acdiflus
acptke
acdifv1
aro_ground_param
aro_ground_diag
arp_ground_param
acdifv2
actkecoefkh
acdifv3
acptkes
acpscc
actkecoefkh
acpblhtm
acpblh
arocldia
acdnsht
acdrag
wrphtrajtm
acpluie
acpluis
actqsats
acnebcond
accdev
accdev
actqsats
accvud
acupu
actqsats
aplmphys
acupm
actqsats
acmodo
acupd
accvimp
accvimp_v3
acpluiz
acuptq
wrscmr
wrscmr
wrscmr
wrscmr
acnebc
qngcor
acdrac
acdrov
acdrme
acozone
acdifoz