

Porting the COSMO v5 Fortran code to GPU

Stefan Rüdüsühli

Swiss COSMO User Workshop

3 December 2014



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

CSCS
Swiss National Supercomputing Centre



COSMO goes GPU: Overview

- Different approaches:
 - Dynamical core: complete re-write (C++)
 - Physics/assimilation/etc.: extend Fortran code (OpenACC)
- Reference: OPCODE (GPU prototype based on COSMO v4.X)
 - Started over with COSMO v5.0
- My areas of work:
 - radiation
 - assimilation
 - output
 - single precision

OpenACC

- Directives: special comments, ignored on CPU
- Original code largely unaltered
- Separate CPU and GPU memory
- Relatively new technique

OpenACC

- Directives: special comments, ignored on CPU
- Original code largely unaltered
- Separate CPU and GPU memory
- Relatively new technique

```
DO k=1,ke
  DO j=1,je
    DO i=1,ie
      p(i,j,k) = p0(k)+pp(i,j,k)
    ENDDO
  ENDDO
ENDDO
PRINT*, MINVAL(p(:, :, ke))
```

OpenACC

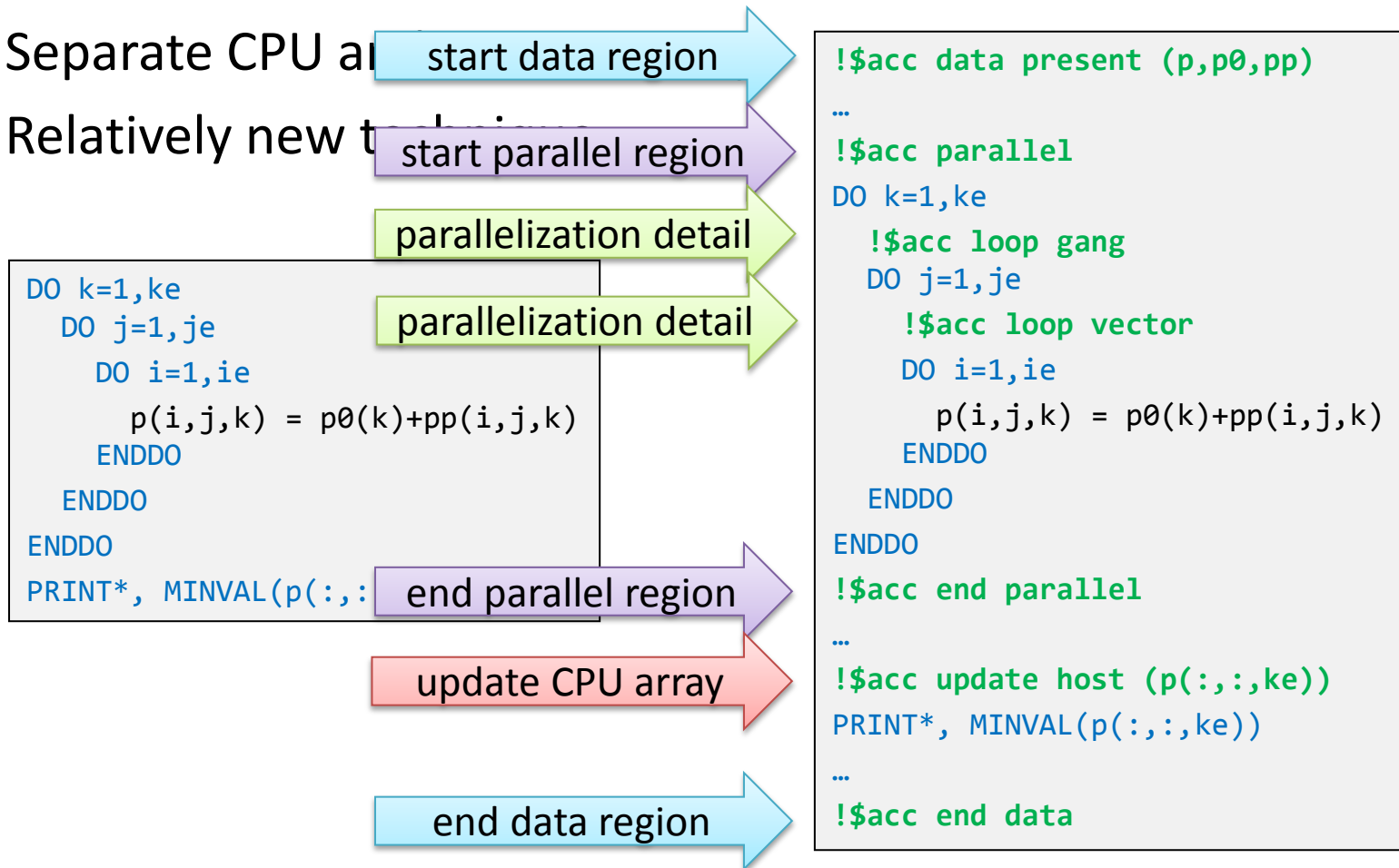
- Directives: special comments, ignored on CPU
- Original code largely unaltered
- Separate CPU and GPU memory
- Relatively new technique

```
DO k=1,ke
  DO j=1,je
    DO i=1,ie
      p(i,j,k) = p0(k)+pp(i,j,k)
    ENDDO
  ENDDO
ENDDO
PRINT*, MINVAL(p(:, :, ke))
```

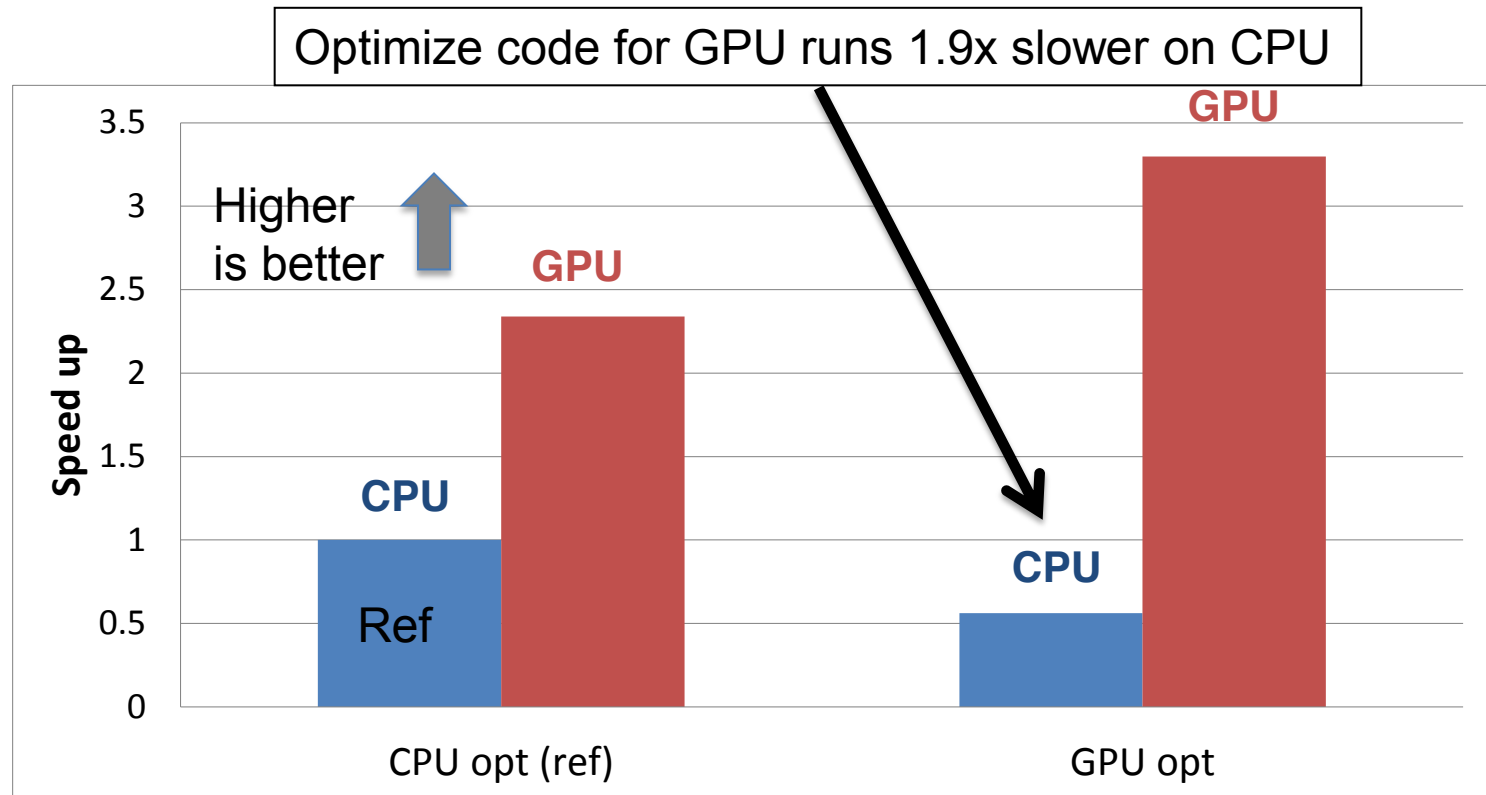
```
!$acc data present (p,p0,pp)
...
!$acc parallel
DO k=1,ke
  !$acc loop gang
  DO j=1,je
    !$acc loop vector
    DO i=1,ie
      p(i,j,k) = p0(k)+pp(i,j,k)
    ENDDO
  ENDDO
ENDDO
!$acc end parallel
...
!$acc update host (p(:, :, ke))
PRINT*, MINVAL(p(:, :, ke))
...
!$acc end data
```

OpenACC

- Directives: special comments, ignored on CPU
- Original code largely unaltered
- Separate CPU and GPU arrays
- Relatively new technology



Radiation performance



- Key kernel of radiation: Speed-up with respect to reference code on CPU
- Radiation is the strongest example, on average in the physics, code optimized for GPU is 1.3x times slower when run on CPU

[Xavier Lapillonne]

Thanks for your attention

```

657
658
659 ! Module procedures in "radiation_rg_tmp"
660
661
662 CONTAINS
663
664
665 ! Module procedure in "radiation_rg_tmp"
666
667
668 SUBROUTINE radiation_rg_organize(
669   ierror      ,yerrmsg      ,ydate_ini  ,lradstep
670   nproma      ,ke           ,ke_soil    ,ke_snow
671 ! INTENT(IN) read-only
672   aer_bc      ,aerlan       ,aer_or     ,aersea  ,aer_ss
673   aer_su      ,aerurb       ,alb_dif    ,alb_dry  ,alb_sat
674   clc_con     ,depth_lk    ,dp0      ,emis_rad ,for_d
675   for_e       ,freshsnow   ,h_ice    ,hmo3     ,llandmask
676   p0          ,p0hl        ,p0cov     ,pp        ,ps
677   qc          ,qi          ,qv         ,rcld     ,soiltyp
678   swdir_cor   ,t           ,t_g        ,t_ice    ,t_s
679   t_snow     ,t_snow_mult ,vio3      ,w_ql     ,w_snow
680   w_so       ,zskyview    ,zsmu0_fesft,zsmu0_flux
681 ! INTENT(INOUT) conditional write-only
682   alb_rad     ,clch        ,clcl       ,clcm     ,clct
683   lwd_s       ,lwu_s       ,qc_rad     ,qi_rad   ,sodwddm
684   thbs       ,thbt        ,thhr
685 ! INTENT(INOUT) read-write
686   aerdes     ,aer_du      ,clc_sgs   ,sod_t    ,sotr
687   sotr_par   ,swtrdifd_s ,swtrdifu_s,swtrdir_s
688 ! INTENT(OUT) unconditional write-only
689   pabs       ,sobs       ,sobt       ,swdifd_s ,swdifu_s
690   swdir_s    ,sohr
691 )
692
693 !
694 ! Description:
695 !
696 ! The module procedure organize_radiations forms the interface between
697 ! the model and the radiation code adapted from the global model gm_e.
698 !
699 ! Method:
700 !
701 ! All variables that are required for the radiation code (i.e. input arrays
702 ! and scalars) are provided or calculated from the model variables.
703 ! The results are stored as solar and thermal heating rates on the
704 ! corresponding global arrays sohr and thhr.
705 !
706 !
707 !
708 ! T.R.:
709 ! Die Variablen zfltf und zflsf (pltf plfs in fesft) wurden entfernt, da sie
710 ! nicht weiterverwendet werden. lcrf und ldebug wurden von Variablen zu
711 ! Parametern.
712 ! Methode fuer Strahlungsrechnung auf groebere Gitter:
713 ! Die Eingangsgroessen fuer die Routine fesft werden auf groebere Gitter
714 ! (Variablenamen der gemittelten Groessen: XXX m).

```

```

5208
5209
5210 !$acc end parallel
5211
5212 ELSEIF ((lrad_dust) .AND. (.NOT. lrad_seas) .AND. (lrad_aero)) THEN
5213
5214
5215 !$acc parallel
5216 !$acc loop gang vector
5217 DO jl = klisc, kilcc
5218   zaeoda(jl) = paeq1(jl,j3)*zaea(kspec,1) + tau_abs_dust(jl,j3,kspec)
5219   + paeq2(jl,j3)*zaea(kspec,2)
5220   + paeq3(jl,j3)*zaea(kspec,3) + tau_abs_aero(jl,j3,kspec)
5221   + paeq4(jl,j3)*zaea(kspec,4)
5222   + paeq5(jl,j3)*zaea(kspec,5)
5223
5224   zaeods(jl) = paeq1(jl,j3)*zaes(kspec,1)*(1.0_dp-zaef(kspec,1)) +
5225   tau_streu_dust(jl,j3,kspec)*(1.0_dp-asy_m_ges(jl,j3,kspec)**2))
5226   + paeq2(jl,j3)*zaes(kspec,2)*(1.0_dp-zaef(kspec,2))
5227   + paeq3(jl,j3)*zaes(kspec,3)*(1.0_dp-zaef(kspec,3)) +
5228   tau_streu_aero(jl,j3,kspec)*(1.0_dp-asy_m_aero(jl,j3,kspec)**2))
5229   + paeq4(jl,j3)*zaes(kspec,4)*(1.0_dp-zaef(kspec,4))
5230   + paeq5(jl,j3)*zaes(kspec,5)*(1.0_dp-zaef(kspec,5))
5231
5232   zzg= ( paeq1(jl,j3)*zaes(kspec,1)*(1.0_dp-zaef(kspec,1))*zaeg(kspec,1) +
5233   tau_streu_dust(jl,j3,kspec)*(1.0_dp-asy_m_ges(jl,j3,kspec)**2))*asy_m_ges(jl,j3,kspec)
5234   + paeq3(jl,j3)*zaes(kspec,3)*(1.0_dp-zaef(kspec,2))*zaeg(kspec,2)
5235   + tau_streu_aero(jl,j3,kspec)*(1.0_dp-zaef(kspec,3))*zaeg(kspec,3) +
5236   + paeq4(jl,j3)*zaes(kspec,4)*(1.0_dp-asy_m_aero(jl,j3,kspec)**2.0_dp))*asy_m_aero(jl,j3,kspec)
5237   + paeq5(jl,j3)*zaes(kspec,5)*(1.0_dp-zaef(kspec,4))*zaeg(kspec,4)
5238   / MAX( zaeods(jl,zepopd)
5239   zaeob(jl) = zldB*(4.0_dp+zzg)/(1.0_dp+zzg)
5240 ENDDO
5241 !$acc end parallel
5242
5243 ELSEIF ((.NOT. lrad_dust) .AND. (.NOT. lrad_seas) .AND. (lrad_aero)) THEN
5244
5245 !$acc parallel
5246 !$acc loop gang vector
5247 DO jl = klisc, kilcc
5248   zaeoda(jl) = paeq1(jl,j3)*zaea(kspec,1)
5249   + paeq2(jl,j3)*zaea(kspec,2)
5250   + paeq3(jl,j3)*zaea(kspec,3) + tau_abs_aero(jl,j3,kspec)
5251   + paeq4(jl,j3)*zaea(kspec,4)
5252   + paeq5(jl,j3)*zaea(kspec,5)
5253
5254   zaeods(jl) = paeq1(jl,j3)*zaes(kspec,1)*(1.0_dp-zaef(kspec,1))
5255   + paeq2(jl,j3)*zaes(kspec,2)*(1.0_dp-zaef(kspec,2))
5256   + paeq3(jl,j3)*zaes(kspec,3)*(1.0_dp-zaef(kspec,3)) +
5257   tau_streu_aero(jl,j3,kspec)*(1.0_dp-asy_m_aero(jl,j3,kspec)**2.0_dp))
5258   + paeq4(jl,j3)*zaes(kspec,4)*(1.0_dp-zaef(kspec,4))
5259   + paeq5(jl,j3)*zaes(kspec,5)*(1.0_dp-zaef(kspec,5))
5260
5261   zzg=( paeq1(jl,j3)*zaes(kspec,1)*(1.0_dp-zaef(kspec,1))*zaeg(kspec,1)
5262   + paeq2(jl,j3)*zaes(kspec,2)*(1.0_dp-zaef(kspec,2))*zaeg(kspec,2)
5263   + paeq3(jl,j3)*zaes(kspec,3)*(1.0_dp-zaef(kspec,3))*zaeg(kspec,3) +
5264   tau_streu_aero(jl,j3,kspec)*(1.0_dp-asy_m_aero(jl,j3,kspec)**2.0_dp))*asy_m_aero(jl,j3,kspec)
5265   + paeq4(jl,j3)*zaes(kspec,4)*(1.0_dp-zaef(kspec,4))*zaeg(kspec,4)
5266   + paeq5(jl,j3)*zaes(kspec,5)*(1.0_dp-zaef(kspec,5))*zaeg(kspec,5) )

```