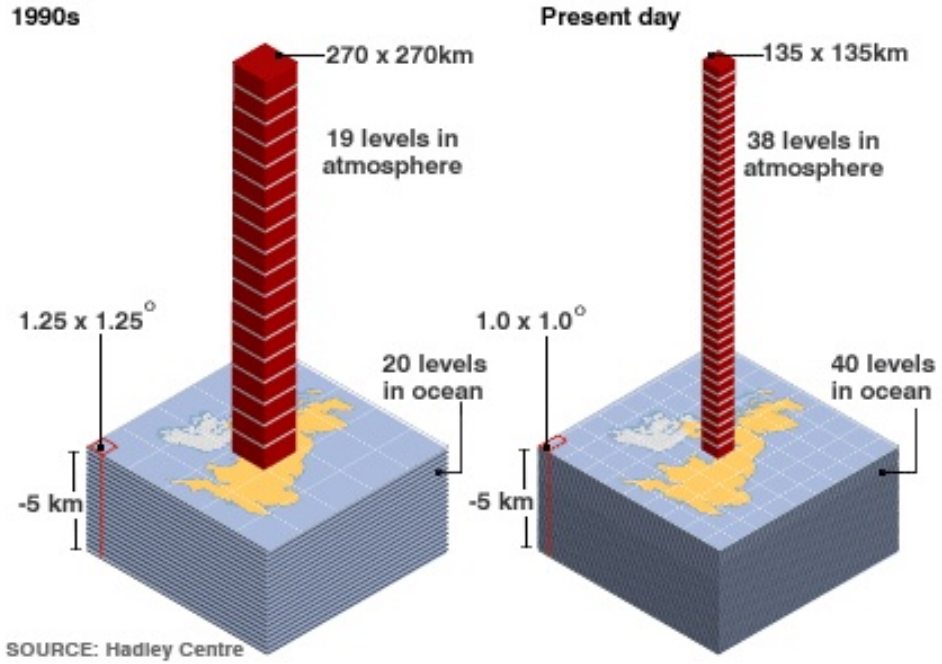
Team ART: aerosol dynamics OpenACC implementation

Will Sawyer, Daniel Rieger,
Markus Wetzstein, Simon Gruber

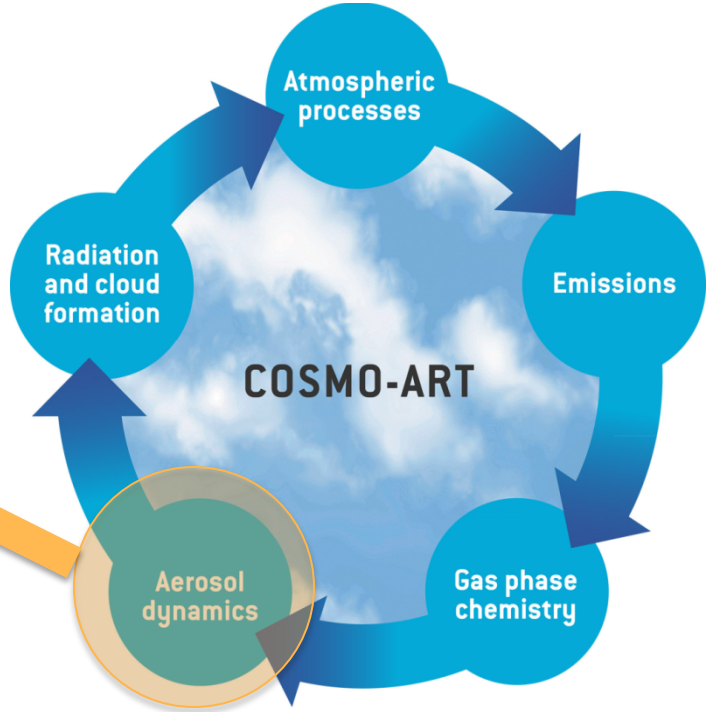
Problem to solve: time loop over columns of data



- CSCS / MeteoSwiss has wanted to port to GPUs for a long time
- “Box model”
- No dependencies between columns
- No dependencies between individual levels in a column
- *Should be perfect for GPUs!*

The Model System COSMO - ART

```
$ wc -l *.f90
    783 art_aerosol_const.f90
     77 art_gas_const.f90
    610 art_gas_const_radm2.f90
  47191 art_isorropia.f90
   9154 art_mademod.f90
    208 art_species_data.f90
    118 data_cosmo_art.f90
     80 data_parameters.f90
    430 made_wrapper.f90
  58651 total
```



~100 !$ACC directives

F77 refactored into F90, strategy for !$ACC ROUTINE

Prior profile

Table 1: Profile by Function Group and Function (top 10 functions shown)

```
  Samp% |   Samp | Imb. |  Imb. |Group
        |        | Samp | Samp% | Function

 100.0% | 141.0 |   -- |    -- |Total
|-----------------------------------------------------
|  66.0% |  93.0 |   -- |    -- |USER
||----------------------------------------------------
||  17.7% |  25.0 |   -- |    -- |made_wrapper_
||  14.2% |  20.0 |   -- |    -- |aerostep$art_mademod_
||  12.8% |  18.0 |   -- |    -- |coagrate$art_mademod_
||   9.9% |  14.0 |   -- |    -- |modpar$art_mademod_
||   5.0% |   7.0 |   -- |    -- |rpmares$art_mademod_
||   2.8% |   4.0 |   -- |    -- |vdvg$art_mademod_
||   1.4% |   2.0 |   -- |    -- |eql3$art_mademod_
||   1.4% |   2.0 |   -- |    -- |nuclcond$art_mademod_
||====================================================
|  34.0% |  48.0 |   -- |    -- |ETC
||----------------------------------------------------
||  22.7% |  32.0 |   -- |    -- |_RTOR_W_00
||   5.0% |   7.0 |   -- |    -- |__cray_dset_SNB
||   1.4% |   2.0 |   -- |    -- |__cray2_EXP_W_00
||   1.4% |   2.0 |   -- |    -- |__cray2_EXP_00
|=====================================================
```

CPU/GPU execution times

Realistic problem size: 64x64 columns each with 50 levels.
360 time steps until data needed on CPU, e.g., for output

Execution time (s) 586.6932730674744

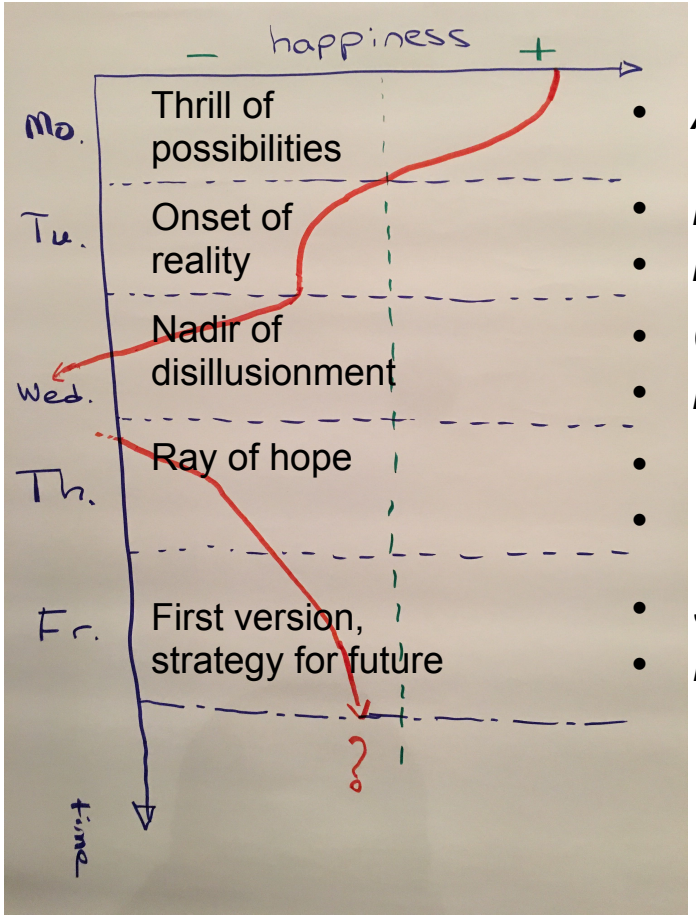
Fair comparison: no OpenMP in code. Assume perfect thread scaling on 16 core node (e.g. 2x8-core Sandybridge) 16x16 cols

Execution time (s) 19.28878498077393

GPU execution: runs on GPU but with numerous UPDATEs to validate results

Execution time (s) 475.1545610427856

ART Hackathon Evolution



- *ART is ideal for GPUs: highly parallel, large index space*
- *K-loop needs to be restructured to extract parallelism*
- *Legacy F77 code needs refactoring for effective use*
- *Code segfaults with CCE, only working with PGI*
- *First PARALLEL loop gives wrong results, hours lost*
- *Workaround for first loop*
- *Whole team adding PARALLEL regions, almost finished*
- *Slowly removing !$ACC UPDATE directives, validating*
- *Initial results…*

Loop of disillusionment

```
!$ACC PARALLEL PRESENT(cgas_new, cblk )
!$ACC LOOP
    DO k = 1, ke
!!!      DO j = 1, iend*jend
      DO jj = 1, jend
        DO ii = 1, iend
!!!        jj = j - (INT((j-1)/jend)*jend)
!!!        ii = INT((j-1)/jend) + jstart
        J = (JJ-1)*jend + II
        DO l = 1, lspec
          cblk(j,k,l) = cgas_new(ii,jj,k,l)
        END DO
        END DO
      END DO
    END DO
!$ACC END PARALLEL
```

Typical PARALLEL region

```
!$ACC PARALLEL LOOP GANG VECTOR COLLAPSE(2)                                  &
!$ACC                 PRESENT(blkdens,blkprs,blkta,cblk,vsulf_in,nitrate_in,nh3_in,   &
!$ACC                       so4rat,so4rat_in,condvap_in,drog,drog_in,epm25i,epm25j, &
!$ACC                       eorgi,eorgj,eeci,eecj,eso4i,eso4j,epm25,eseas)
   DO kk= 1, ke
     DO  ncell=1,imax*jmax
       blkdens(ncell,kk)    = blkprs(ncell,kk)/(rdgas*blkta(ncell,kk)) !dry air density[kg m**-3]
       cblk(ncell,kk,vsulf) = vsulf_in(ncell,kk)
       cblk(ncell,kk,vhno3) = nitrate_in(ncell,kk)
       cblk(ncell,kk,vnh3)  = nh3_in(ncell,kk)
   :
     END DO !ncell
   END DO !k
!$ACC END PARALLEL
```

Outlook on isoropia (F77 legacy)

ca. 50k lines of code

called in an already parallel loop -> needs `routine seq`

F77 version cannot currently be ported to OpenACC: common blocks and block data initialization -> needs to be ported with `declare copy` BUT currently (PGI16.7) these can't be in an `acc routine` (only directly in a parallel region)

good news: F90 version exists,still needs to be ported, but strategy is clear

OpenACC comments

1. Loop of disillusionment: PGI 16.7 may generate incorrect code
2. Cray `ftn` wrapper changes default behavior of PGI compiler
3. CCE 8.4.6 does not implement OpenACC2.5, e.g., `DEFAULT(PRESENT)`
4. Slight differences between CCE and PGI OpenACC, e.g., `!$ACC PARALLEL LOOP` can be terminated by `!$ACC END PARALLEL` (PGI) but with CCE must be `!$ACC END PARALLEL LOOP`

What we learned

- Using GIT on a highly collaborative project
- Learning OpenACC, essentially from scratch, now have the tools to port rest of code
- Code required major refactoring to benefit from GPU
- In spite of initial performance we believe GPU node can be faster than 1 CPU node (2 sockets)
- We hope to continue until we reach that point