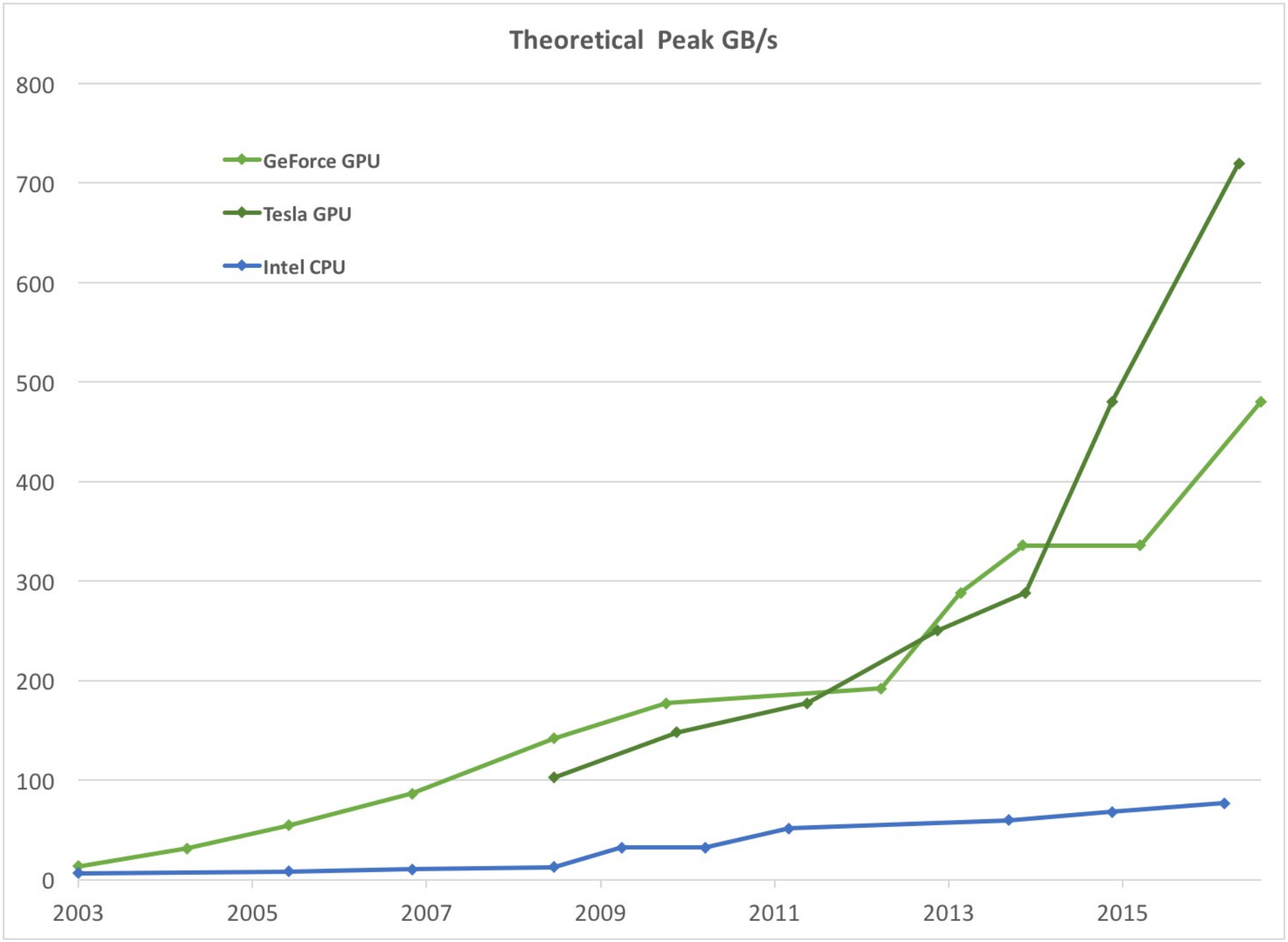
What should we expect from GPUs ?

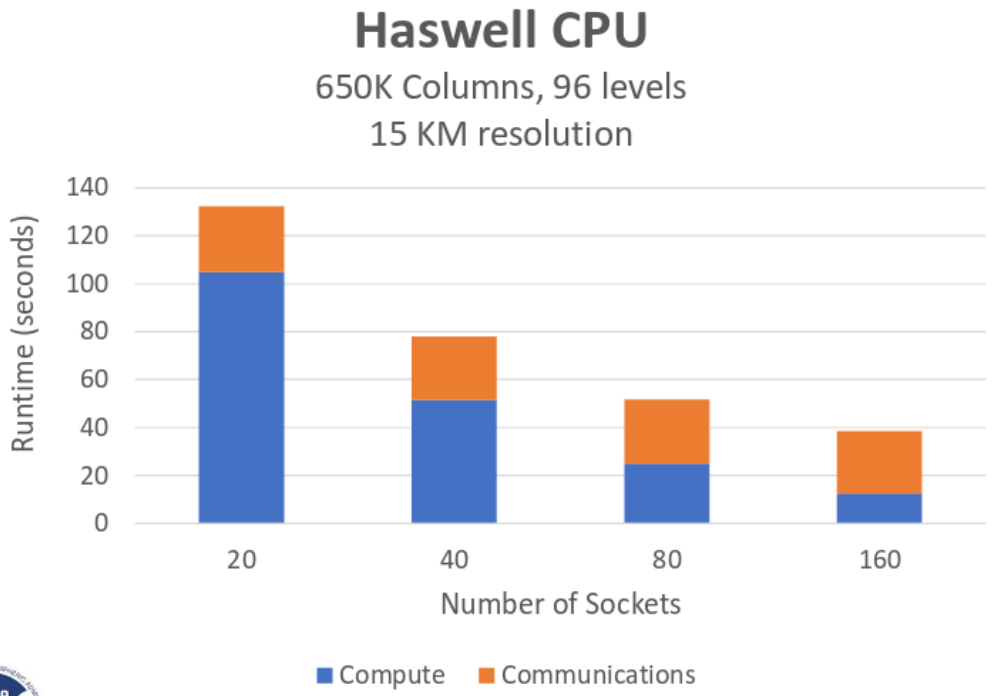
- Introduction
- Profitability of GPUs
- Memory management
- Porting the physics
- Running & comparing test cases

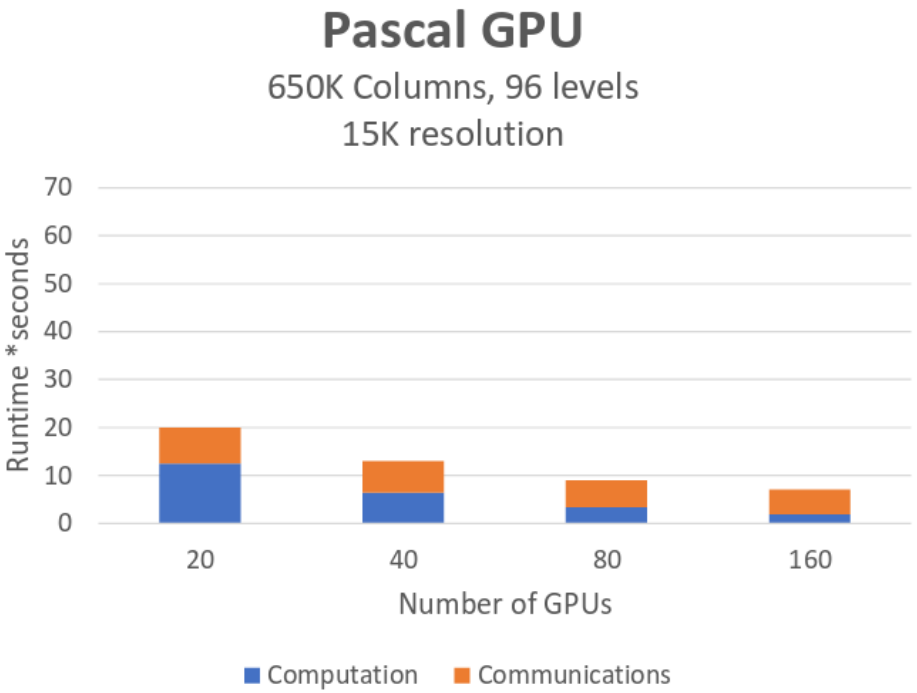

Theoretical Peak GB/s
800
700
600
500
400
300
200
100
0
GeForce GPU
Tesla GPU
Intel CPU
2003
2005
2007
2009
2011
2013
2015

Porting NOAA NWP model on GPUs

Strong Scaling: CPU, GPU

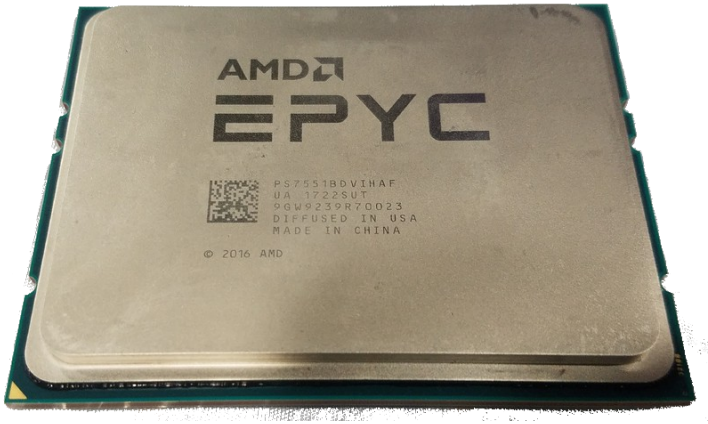
- CPU socket (2 per node) versus Pascal GPU (2 per node)
  - Identical system, interconnect, data movement per MPI task
  - Different communications CPU: impi, GPU: mvapich), affinity (CPU: range, GPU: pinned)





7TFlops
900Gb/s

1TFlops
170Gb/s



700 horsepower
400 km/h

45 horsepower
120 km/h

	NVIDIA TESLA K80	Broadwell E5-2698 v4 2.20GHz
Peak performance double prec. (GFlops)	1870	704
Memory bandwidth (Gb/s)	480	76
Availability	November 2014	June 2015
Power (W)	300	135
Price (€)	4 237	2889 + ????
Memory (Gb)	24	32
GFlops/€	0,44	0,24
Gb/s/€	0,11	0,02

	NVIDIA TESLA V100	AMD Naples EPYC 7551P
Peak performance double prec. (GFlops)	7000	1024
Memory bandwidth (Gb/s)	900	170
Availability	August 2017	March 2017
Power (W)	300	180
Price (€)	7600	2500 + 1600 = 4100
Memory (Gb)	32	64
GFlops/€	0,92	0,41 ; 0,24
Gb/s/€	0,12	0,07 ; 0,04

Question 1 to NVIDIA

AMD Naples vs TESLA V100 : do you think that a reduction of a factor 2 to 3 of the total cost is possible ?

Answer : no answer

AROME 1.3km on Broadwell

- 180 Broadwell nodes (= 360 sockets)
- 2160 time steps in 1800s, (0.8s per time step)
- Power consumption of a Broadwell node : idle = 30W, running = 300W
- Mensual fee for two 1800 node Broadwell clusters = 500k€
- Cost of kWh (in France) : 0,15€

AROME 1.3km on Broadwell

Cost of a single time step :

- Fixed cost (hardware & vendor support) :
  0,0078 €
- Energy cost : 0,0017 €

Time step of AROME

- Inverse spectral transform
- Grid-point calculation (physics)
- Semi-lagrangian
- Direct spectral transform
- Semi-implicit calculations
- Inverse spectral transform
- Direct spectral transform
- Semi-implicit calculations

Porting a part of AROME to GPU ?

Example: Port 50 % of the code (total time) on a GPU (same price as a CPU), with a speedup factor of 2 :

→ fixed cost x 2

→ total speedup on the application = 1,33

Cost model

- α : fraction of the code running on CPU, hence (1 - α) is the fraction running on GPU.
- $\beta > 1$: speedup on GPU vs CPU
- N_C : number of CPUs
- N_G : number of GPUs
- C_H : cost of hardware = $2{,}7 \times 10^{-5}$ €/s
- C_E : cost of energy = 4×10^{-8} €/J
- P_0 : power consumption of an idle processor = 15W
- dP : power consumption of a busy processor = 135W
- T : elapsed time of an AROME time step = 0.8s
- μ : T x 360 = 288 ; 360 is the number of Broadwell sockets used by AROME in order to meet the operational constraint (30h/1800s)

Cost model

Cost of a time step on a CPU cluster :

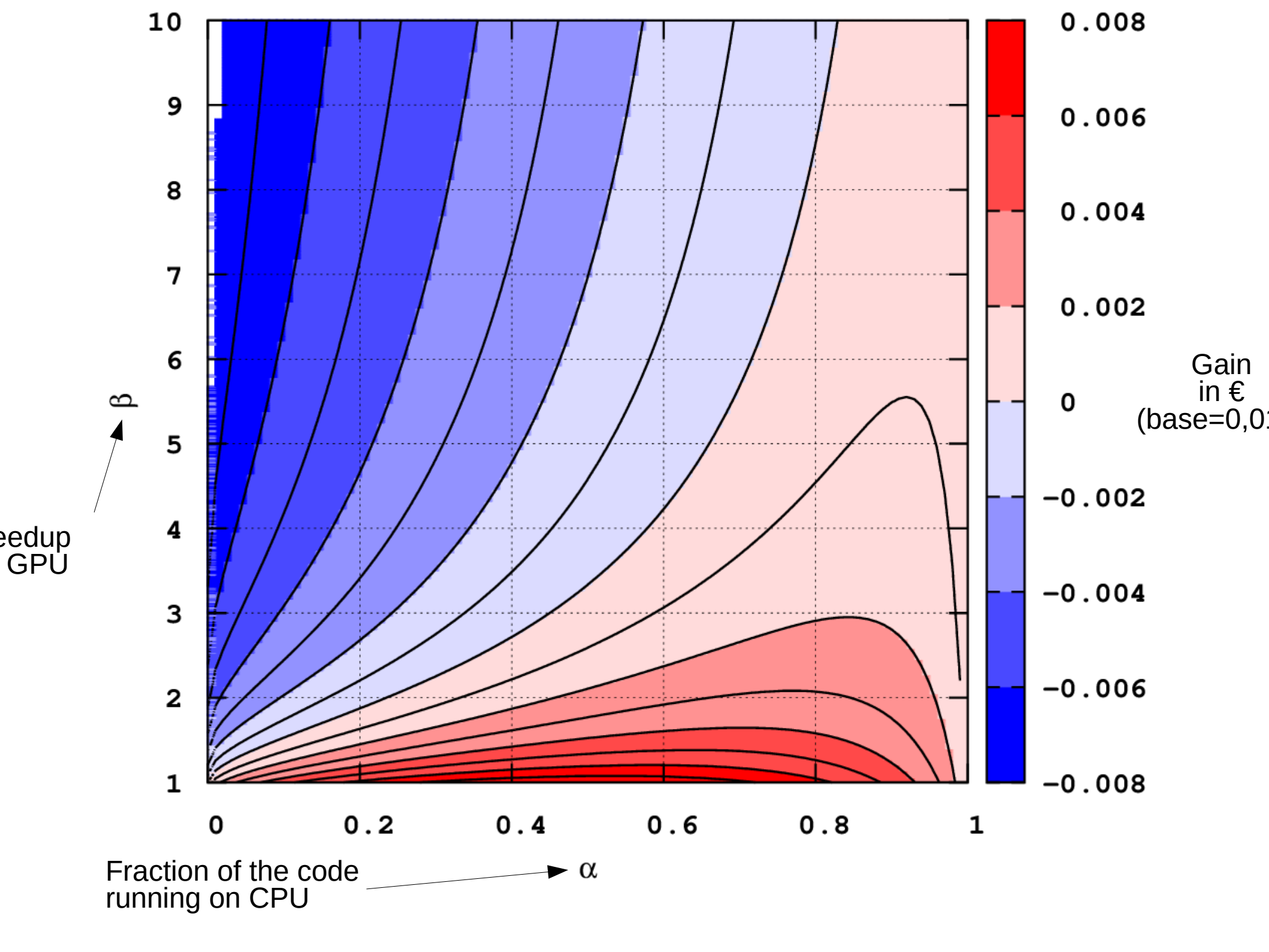
$$C(E)\times T\times dP(CPU)\times NCPU + C(E)\times T\times CPU\times P0(CPU) + NCPU\times Cost(CPU)\times T$$

Cost of a time step on CPU+GPU cluster :

$$C(E)\times TGPU\times dP(GPU)\times NGPU_B + C(E)\times(T-TGPU)\times dP(CPU)\times NCPU_B$$

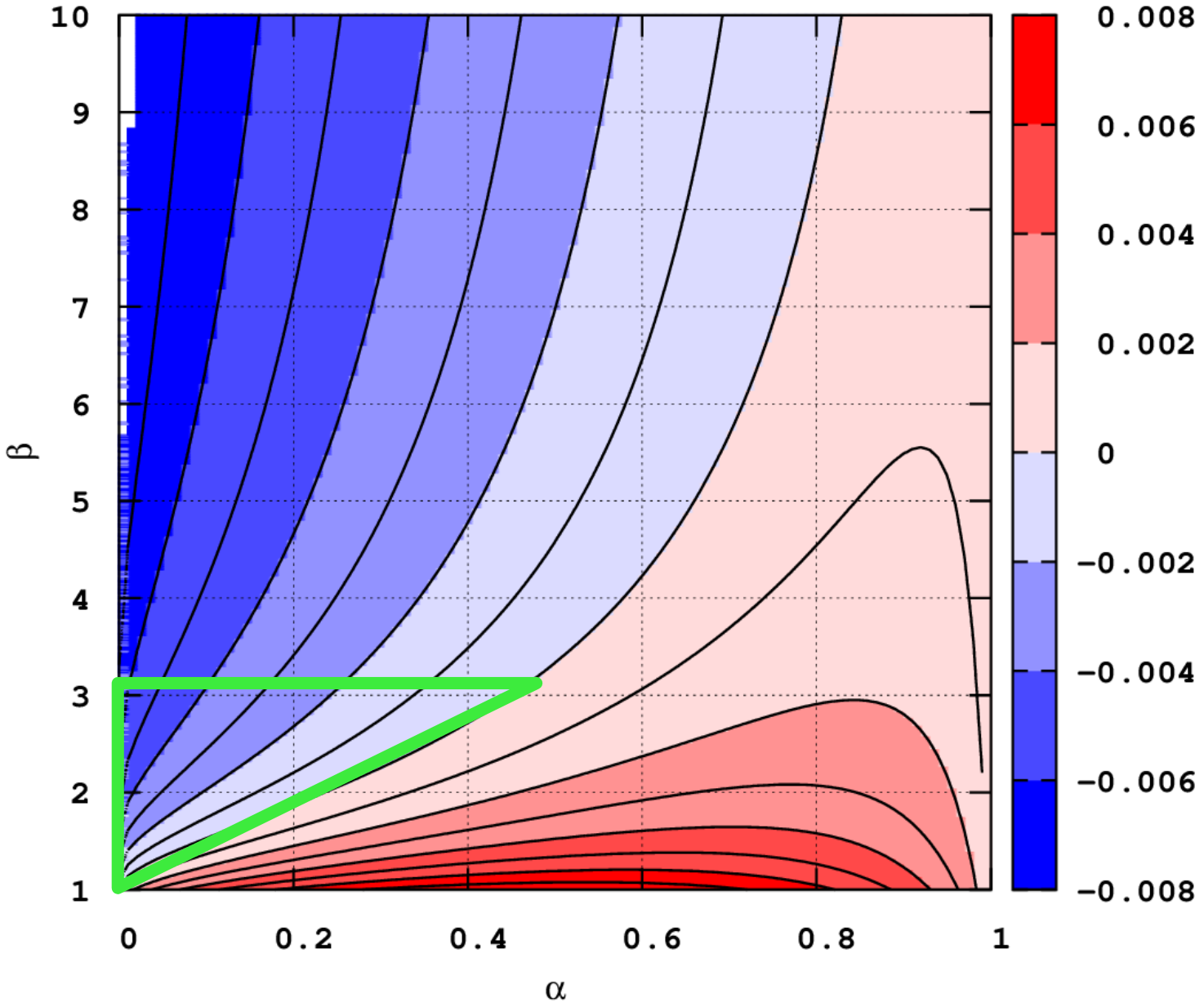
$$+ C(E)\times T\times(NGPU_B\times P0(GPU) + NCPU_B\times P0(GPU))$$

$$+ (NGPU_B\times Cost(GPU) + NCPU_B\times Cost(CPU))\times T$$


10
9
8
7
6
5
4
3
2
1
β
eedup
GPU
0
0.2
0.4
0.6
0.8
1
α
Fraction of the code
running on CPU
0.008
0.006
0.004
0.002
0
−0.002
−0.004
−0.006
−0.008
Gain
in €
(base=0,0

70 % of the code on GPU
→ 20 % on TCO

90 % of the code on GPU
→ 40 % on TCO



Question 2 to NVIDIA

Assuming that the gain on GPU is 3, does porting 90 % of the code on GPU in order to reduce the TCO by a factor of 2 sound reasonnable ?

Answer : Yes

Memory

- No stack management (for now) :
  - All automatic arrays have to be allocated by the user, outside of GPU code
  - Memory overhead

- AROME : 1440Gb (measured)
  - 360 Broadwell sockets (5Tb of memory)
  - ~ 60 TESLA K80 (factor of 6 on memory bandwidth) ; 24Gb x 60 = 1440 Gb

Question 3 to NVIDIA

How do we manage a stack on GPU ?

Do you think AROME can fit on GPUs ?

Answers : progress on stack management, but that would not solve the memory overhead.

Having AROME on GPU would imply a strong effort, just to fit inside the small memory

Physics

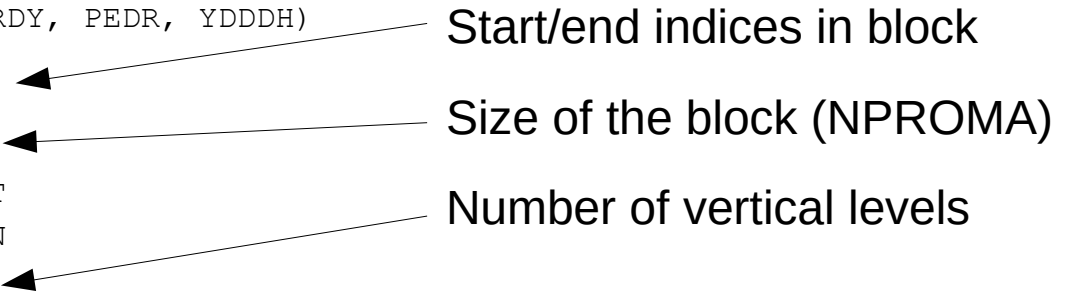
```
REAL(KIND=JPRB) :: ZRT0M(NPROMA,NFLEVG,NGPBLKS)

REAL(KIND=JPRB) :: ZXYB0(NPROMA,NFLEVG,NDIM,NGPBLKS)

REAL(KIND=JPRB) :: ZUVH0(NPROMA,0:NFLEVG,NDIM,NGPBLKS)
```

Physics

```
!$OMP PARALLEL DO SCHEDULE(DYNAMIC,1) PRIVATE (JKGLO,IBL)

DO JKGLO=1,KGPCOMP,NPROMA

  IBL=(JKGLO-1)/NPROMA+1

  CALL CPG(YDGEOMETRY,YDGMV,YDSURF, &
    & LLCONFX,LD_DFISTEP,LDRETCFOU,LDWRTCFOU0,LLFSTEP,LLRESET,LLDIAB,LLSLPHY,LLUSEPB1,&
    & KGPCOMP,JKGLO,IBL,YDOROG(IBL),&
    & ZDT,ZDTPHY,ZTE,ZBETADT,YDSL,PGFLSLP(1,1,1,IBL),PSAVTEND(1,1,1,IBL),&
    & PGMVTNDHD_DDH(1,1,1,IBL),PGFLTNDHD_DDH(1,1,1,IBL),PGPSDT2D(1,1,IBL),PSD_PF(1,1,1,IBL),&
    & PGMV(1,1,1,IBL),PGMVS(1,1,IBL),PGFL(1,1,1,IBL),PGFLPC(1,1,1,IBL),PGFLPT(1,1,1,IBL),&
    & PSP_SB(1,1,1,IBL),PSP_SG(1,1,IBL),PSP_RR(1,1,IBL),PSD_VF(1,1,IBL),PSD_VP(1,1,IBL),&
    & PSD_VV(1,1,IBL),PSD_VH(1,1,IBL),PSD_VA(1,1,IBL),PSD_VC(1,1,IBL),PSD_VD(1,1,IBL),&
    & PSD_SFL(1,1,IBL),PSD_SFO(1,1,IBL),PSD_XA(1,1,1,IBL),PSD_DI(1,1,1,IBL),&
    & PEMTD(1,1,IBL),PEMTU(1,1,IBL),PTRSW(1,1,IBL),PRMOON(1,IBL),PGMVTNDSI_DDH(1,1,1,IBL),&
    & PGDEOSI(1,0,1,IBL),PGUEOSI(1,0,1,IBL),PGMU0(1,1,IBL),PGMU0_MIN(1,IBL),PGMU0_MAX(1,IBL),&
    & PGDEOTI(1,0,IBL),PGDEOTI2(1,0,IBL),PGUEOTI(1,0,IBL),&
    & PGUEOTI2(1,0,IBL),PGEOLT(1,1,IBL),PGEOXT(1,1,IBL),&
    & PGRPROX(1,0,IBL),PGMIXP(1,0,IBL),PGFLUXC(1,0,IBL),PGRSURF(1,IBL),&
    & IWSETTLOFF(1,IBL),PB1,PB2(1,1,IBL),PGMVT1(1,1,1,IBL),PGMVT1S(1,1,IBL),PGFLT1(1,1,1,IBL),PEXTRA(1,1,1,IBL),&
    & PGMVTNDSL_DDH(1,1,1,IBL),PGFLTNDSL_DDH(1,1,1,IBL),PTRAJEC%PHYS(IBL),&
    & PTRAJEC%SLAG(IBL))

ENDDO

!$OMP END PARALLEL DO
```

Physics

```
SUBROUTINE ACTKE ( KIDIA, KFDIA, KLON, KTDIAT, KTDIAN, KLEV,&
            & PAPHI, PAPHIF, PAPRS, PAPRSF, PDELP, PR, PT,&
            & PU, PV, PQ, PLSCPE, &
...
            & PNEBS, PQCS, PNEBS0, PQCS0, PCOEFN , &
            & PFECT , PFECTI, PECT1 , PTPRDY, PEDR, YDDDH)
INTEGER(KIND=JPIM),INTENT(IN)    :: KIDIA
INTEGER(KIND=JPIM),INTENT(IN)    :: KFDIA
INTEGER(KIND=JPIM),INTENT(IN)    :: KLON
INTEGER(KIND=JPIM),INTENT(IN)    :: KTDIAT
INTEGER(KIND=JPIM),INTENT(IN)    :: KTDIAN
INTEGER(KIND=JPIM),INTENT(IN)    :: KLEV
REAL(KIND=JPRB)   ,INTENT(IN)    :: PAPHI(KLON,0:KLEV)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PAPHIF(KLON,KLEV)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PAPRS(KLON,0:KLEV)
REAL(KIND=JPRB)   ,INTENT(IN)    :: PAPRSF(KLON,KLEV)
...
REAL(KIND=JPRB) :: ZUSLE  (KLON,KLEV), ZLMECT(KLON,KLEV), ZPHI3(KLON,KLEV)
REAL(KIND=JPRB) :: ZPRODTH(KLON,KLEV),ZPRDY(KLON,KLEV)
REAL(KIND=JPRB) :: ZDIAG(KLON,KLEV)
REAL(KIND=JPRB) :: ZDIFF(KLON,KLEV),ZDISS(KLON,KLEV)
REAL(KIND=JPRB) :: ZECT(KLON,KLEV), ZECT1(KLON,KLEV)
```



Style(s)

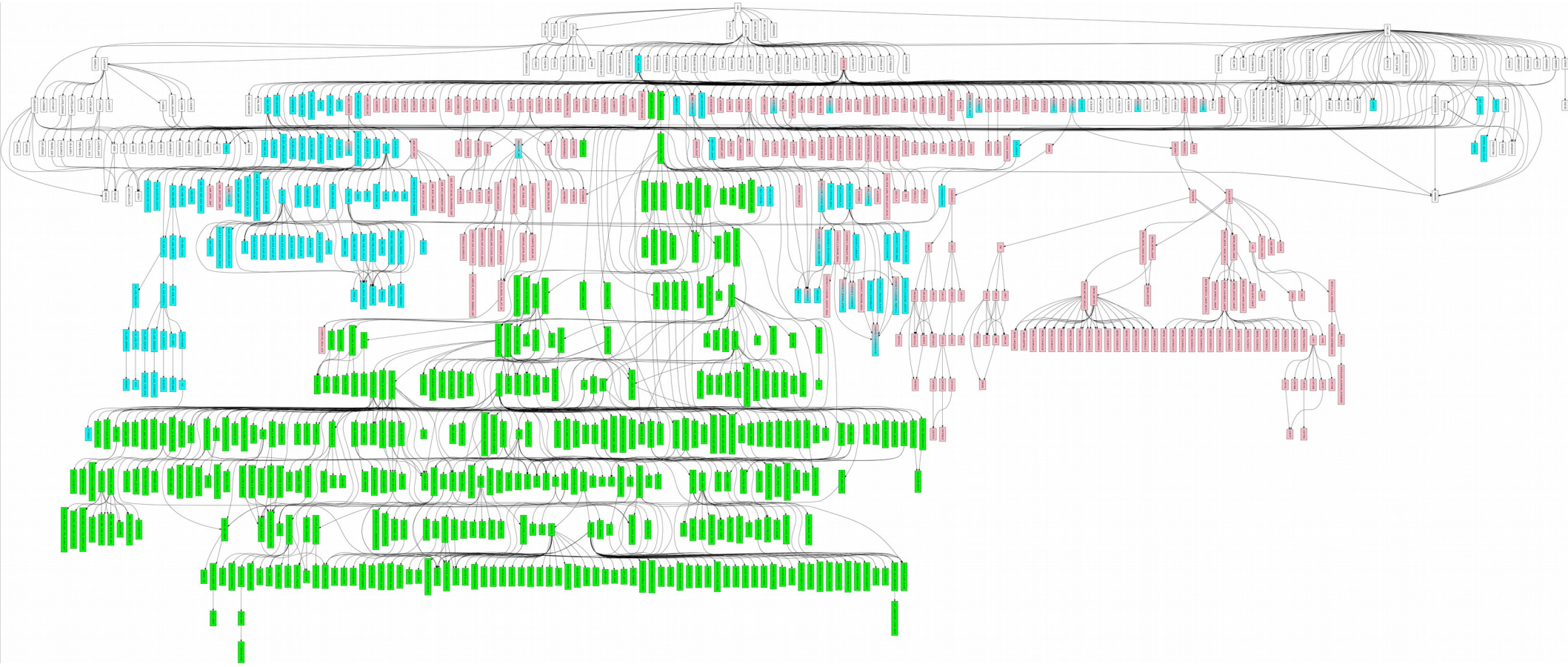
« ARPEGE »

```
DO JLEV=KTDIAN,KLEV
  DO JLON=KIDIA,KFDIA
     ZGZTOP(JLON) = MAX(ZGZTOP(JLON),PAPHIF(JLON,JLEV)*PNLAB(JLON,JLEV))
     ZGZBOT(JLON) = (1.0_JPRB-PNLAB(JLON,JLEV))*ZGZBOT(JLON) &
    &             + PNLAB(JLON,JLEV)*MIN(ZGZBOT(JLON),PAPHIF(JLON,JLEV))
  ENDDO
ENDDO
```

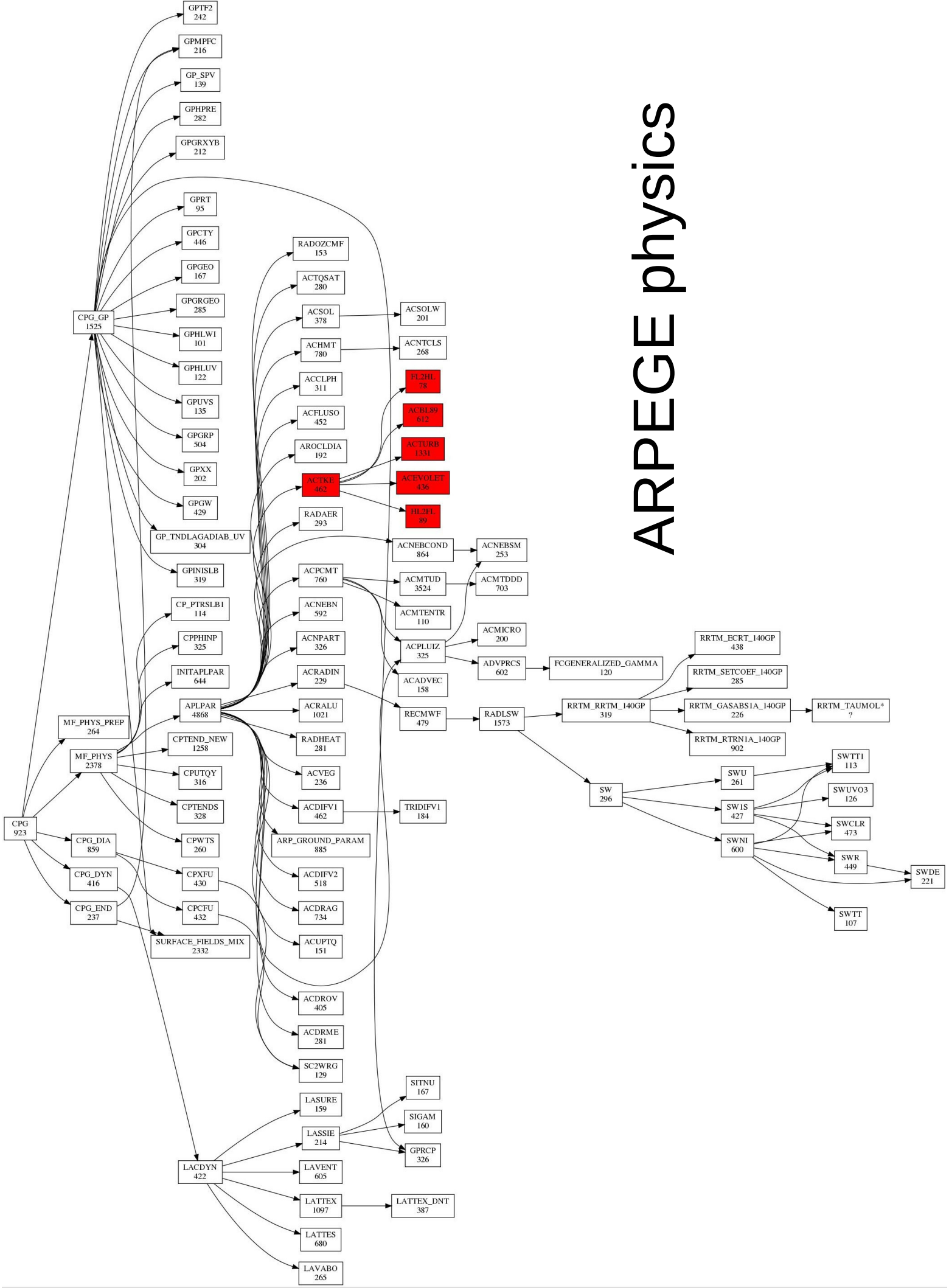
« AROME/MesoNH »

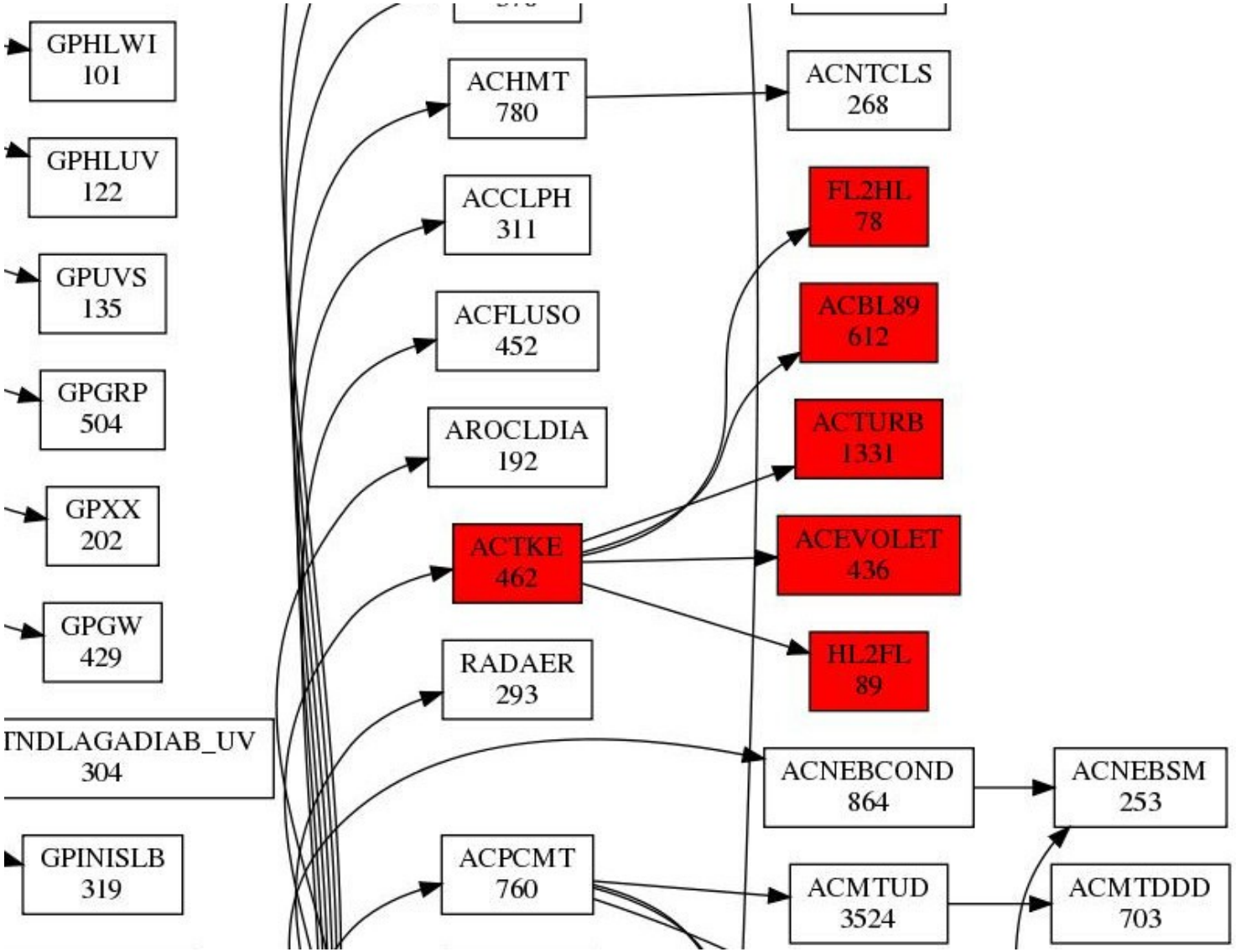
```
ZRAY(:,:,:)   = 0.
ZCONC_TMP(:,:)=PSEA(:,:)*XCONC_SEA+(1.-PSEA(:,:))*XCONC_LAND

DO JK=IKTB,IKTE
  ZLBC(:,:,JK)   = PSEA(:,:)*XLBC(2)+(1.-PSEA(:,:))*XLBC(1)
  ZFSEDC(:,:,JK) = (PSEA(:,:)*XFSEDC(2)+(1.-PSEA(:,:))*XFSEDC(1))
  ZFSEDC(:,:,JK) = MAX(MIN(XFSEDC(1),XFSEDC(2)),ZFSEDC(:,:,JK))
  ZCONC3D(:,:,JK)= (1.-PTOWN(:,:))*ZCONC_TMP(:,:)+PTOWN(:,:)*XCONC_URBAN
  ZRAY(:,:,JK)   = 0.5*((1.-PSEA(:,:))*GAMMA(XNUC+1.0/XALPHAC)/(GAMMA(XNUC)) + &
  PSEA(:,:)*GAMMA(XNUC2+1.0/XALPHAC2)/(GAMMA(XNUC2)))
END DO

WHERE (ZRCT(:).GT.0. .AND. ZCF(:).GT.0.)
  ZHLC_RCMAX(:) = ZCOEFFRCM * ZRCT(:) / ZCF(:)
END WHERE
```

ARPEGE/AROME physics



Green = SURFEX, Pink = ARPEGE, Blue = AROME


ARPEGE physics
GPTF2 242
GPMPFC 216
GP_SPV 139
GPHPRE 282
GPGRXYB 212
GPRT 95
GPCTY 446
GPGEO 167
GPGRGEO 285
GPHLWI 101
GPHLUV 122
GPUVS 135
GPGRP 504
GPXX 202
GPGW 429
CPG_GP 1525
GP_TNDLAGADIAB_UV 304
GPINISLB 319
CP_PTRSLB1 114
CPPHINP 325
INITAPLPAR 644
APLPAR 4868
MF_PHYS_PREP 264
MF_PHYS 2378
CPTEND_NEW 1258
CPUTQY 316
CPTENDS 328
CPWTS 260
CPG 923
CPG_DIA 859
CPG_DYN 416
CPG_END 237
CPXFU 430
CPCFU 432
SURFACE_FIELDS_MIX 2332
RADOZCMF 153
ACTQSAT 280
ACSOL 378
ACHMT 780
ACCLPH 311
ACFLUSO 452
AROCLDIA 192
ACTKE 462
RADAER 293
ACPCMT 760
ACNEBN 592
ACNPART 326
ACRADIN 229
ACRALU 1021
RADHEAT 281
ACVEG 236
ACDIFV1 462
ARP_GROUND_PARAM 885
ACDIFV2 518
ACDRAG 734
ACUPTQ 151
ACDROV 405
ACDRME 281
SC2WRG 129
LACDYN 422
LASURE 159
LASSIE 214
LAVENT 605
LATTEX 1097
LATTES 680
LAVABO 265
ACSOLW 201
ACNTCLS 268
FL2HL 78
ACBL89 612
ACTURB 1331
ACEVOLET 436
HL2FL 89
ACNEBCOND 864
ACNEBSM 253
ACMTUD 3524
ACMTDDD 703
ACMTENTR 110
ACPLUIZ 325
ACMICRO 200
ADVPRCS 602
FCGENERALIZED_GAMMA 120
ACADVEC 158
RECMWF 479
RADLSW 1573
RRTM_RRTM_140GP 319
RRTM_ECRT_140GP 438
RRTM_SETCOEF_140GP 285
RRTM_GASABS1A_140GP 226
RRTM_TAUMOL* ?
RRTM_RTRN1A_140GP 902
SW 296
SWU 261
SW1S 427
SWNI 600
SWTT1 113
SWUVO3 126
SWCLR 473
SWR 449
SWDE 221
SWTT 107
TRIDIFV1 184
SITNU 167
SIGAM 160
GPRCP 326
LATTEX_DNT 387


GPHLWI
101
GPHLUV
122
GPUVS
135
GPGRP
504
GPXX
202
GPGW
429
TNDLAGADIAB_UV
304
GPINISLB
319
ACHMT
780
ACCLPH
311
ACFLUSO
452
AROCLDIA
192
ACTKE
462
RADAER
293
ACPCMT
760
ACNTCLS
268
FL2HL
78
ACBL89
612
ACTURB
1331
ACEVOLET
436
HL2FL
89
ACNEBCOND
864
ACNEBSM
253
ACMTUD
3524
ACMTDDD
703

Principles

- Try to use « ARPEGE » style
- Try to keep the structure of the code
- Minimize particular cases
- Work out general rules to be applied for migration and future coding
- Respect memory coalescing, minimize divergences, maximize instruction parallelism

CPU : OpenMP

```
!$OMP PARALLEL DO PRIVATE (IBLOCK, IIDIA, IFDIA)
DO IBLOCK = 1, NGPBLKS
  IIDIA = 1
  IFDIA = KLON
  CALL SIMPLE4_ACTKE (IIDIA, IFDIA, KLON, KTDIAT, KTDIAN, KLEV, PAPHI (:,:,       &
  & IBLOCK), PAPHIF (:,:, IBLOCK), PAPRS (:,:, IBLOCK), PAPRSF (:,:, IBLOCK),     &
  & PDELP (:,:, IBLOCK), PR (:,:, IBLOCK), PT (:,:, IBLOCK), PU (:,:, IBLOCK),    &
  & PV (:,:, IBLOCK), PQ (:,:, IBLOCK), PLSCPE (:,:, IBLOCK), PCD (:, IBLOCK),    &
  & PCH (:, IBLOCK), PGZ0 (:, IBLOCK), PTS (:, IBLOCK), PQS (:, IBLOCK), PQICE (  &
  & :,:, IBLOCK), PQLI (:,:, IBLOCK), PECT (:,:, IBLOCK), PPRODTH (:,:, IBLOCK),  &
  & PNLAB (:,:, IBLOCK), PNLABCVP (:,:, IBLOCK), PKTROV (:,:, IBLOCK), PKUROV (:, &
  & :, IBLOCK), PXTROV (:,:, IBLOCK), PXUROV (:,:, IBLOCK), PNBVNO (:,:, IBLOCK), &
  & PNEBS (:,:, IBLOCK), PQCS (:,:, IBLOCK), PNEBS0 (:,:, IBLOCK), PQCS0 (:,:,    &
  & IBLOCK), PCOEFN (:,:, IBLOCK), PFECT (:,:, IBLOCK), PECT1 (:,:, IBLOCK),      &
  & PTPRDY (:,:, IBLOCK), PEDR (:,:, IBLOCK), RG, RPRTH, ECTMIN, TSPHY, ACBRPHIM, &
  & ADISE, ADISI, AKN, ALD, ALMAV, ALMAVE, ALMAVX, ALPHAE, ALPHAT, ARSB2, ARSC1,  &
  & ECTMAX, EDB, EDC, EDD, RALPD, RALPS, RALPW, RATM, RBETD, RBETS, RBETW, RCPV,  &
  & RCS, RCW, RD, RDT, RETV, RGAMD, RGAMS, RGAMW, RKAPPA, RLSTT, RLVTT, RTT, RV,  &
  & UDECT, UPRETMAX, UPRETMIN, USHEARM, USURIC, VKARMN, KSIZST, KPTRST, PSTACK (  &
  & :,:, IBLOCK))
ENDDO
!$OMP END PARALLEL DO
```

GPU : Fortran CUDA

```
  IBLOCK = BLOCKIDX%X
  IIDIA  = THREADIDX%X
  IFDIA  = THREADIDX%X
  CALL SIMPLE4_ACTKE (IIDIA, IFDIA, KLON, KTDIAT, KTDIAN, KLEV, PAPHI (:,:,       &
  & IBLOCK), PAPHIF (:,:, IBLOCK), PAPRS (:,:, IBLOCK), PAPRSF (:,:, IBLOCK),     &
  & PDELP (:,:, IBLOCK), PR (:,:, IBLOCK), PT (:,:, IBLOCK), PU (:,:, IBLOCK),    &
...
  & ADISE, ADISI, AKN, ALD, ALMAV, ALMAVE, ALMAVX, ALPHAE, ALPHAT, ARSB2, ARSC1,  &
  & ECTMAX, EDB, EDC, EDD, RALPD, RALPS, RALPW, RATM, RBETD, RBETS, RBETW, RCPV,  &
  & RCS, RCW, RD, RDT, RETV, RGAMD, RGAMS, RGAMW, RKAPPA, RLSTT, RLVTT, RTT, RV,  &
  & UDECT, UPRETMAX, UPRETMIN, USHEARM, USURIC, VKARMN, KSIZST, KPTRST, PSTACK (  &
  & :,:, IBLOCK))

…

CALL RUN_SIMPLE4_ACTKE <<<NGPBLKS, KLON>>> &
&               (KIDIA, KFDIA, KLON, KTDIAT, KTDIAN, KLEV,      &
& PAPHI, PAPHIF, PAPRS, PAPRSF, PDELP, PR, PT, PU, PV, PQ,      &
& PLSCPE, PCD, PCH, PGZ0, PTS, PQS, PQICE, PQLI, PECT,          &
& PPRODTH, PNLAB, PNLABCVP, PKTROV, PKUROV, PXTROV, PXUROV,     &
…
& RGAMD, RGAMS, RGAMW, RKAPPA, RLSTT, RLVTT, RTT, RV, UDECT,    &
& UPRETMAX, UPRETMIN, USHEARM, USURIC, VKARMN, KSIZST, KPTRST, &
& PSTACK, NGPBLKS)
```

Question 4 to NVIDIA

What recommandations would you issue to port our physics to GPU ?

Answer : the NPROMA paradigm looks good ; try to port other bits of the physics.

Spectral transforms dwarf

- Port of spectral transforms to GPU
- Made by NVIDIA
- ESCAPE product
- Does not rely on Atlas
- Spherical harmonics only
- No fast Legendre transforms
- TL1279

Comparison of AMD Naples & NVIDIA V100

- Use actke & spectral transforms dwarf
- AMD EPYC 7601, 2.2GHz, RAM 2400MHz → 150Gb/s
- NVIDIA GV100 850 Gb/s

Spectral transforms

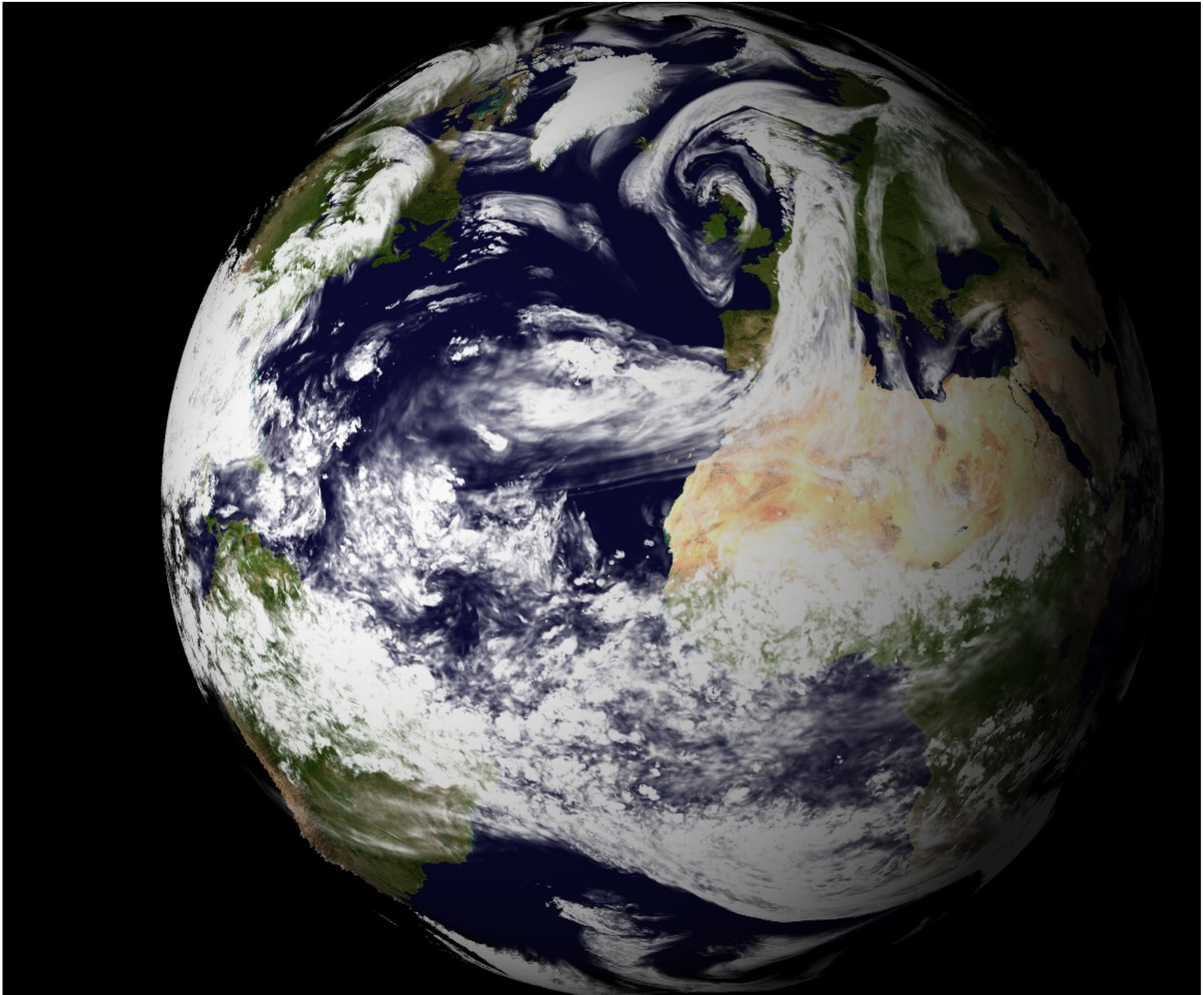
	Time per iteration	Memory	Setup	Price
V100	0,72s	60Gb	55s	7100€
AMD	1,84s	20Gb	4s	4600€

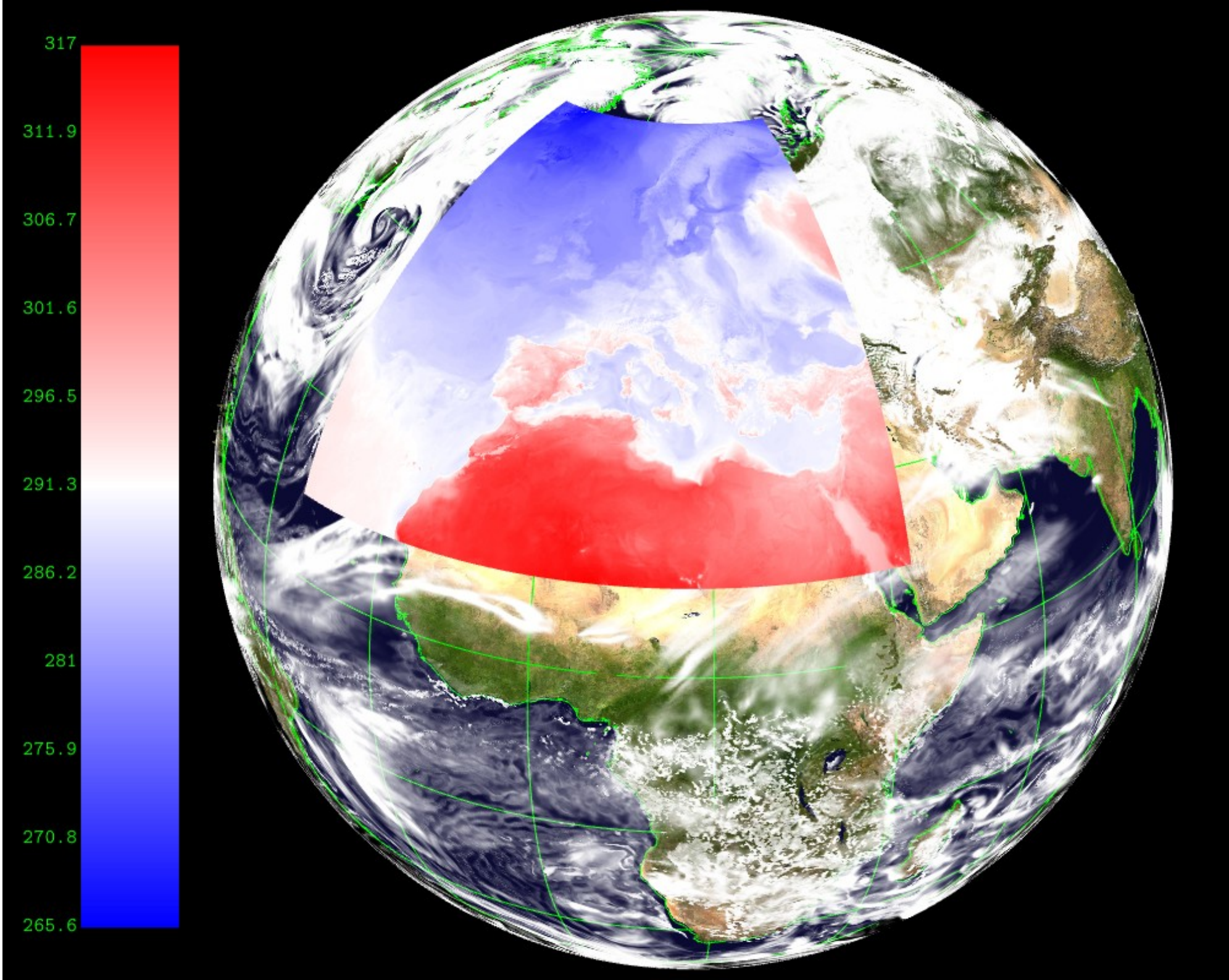
ACTKE

	Time per iteration	Memory	Setup	Price
V100	8s	6,2Gb	29s	7100€
AMD	32s	3,7Gb	3s	4600€

Conclusion & plans

- Profitability (€) of GPU vs CPU to be estimated using other parts of the code
- Take setup and non portable parts into account
- Solve memory management issues
- Work out a global strategy for porting our physics
- Follow developments made at ECMWF for the dynamics
- Look at (quite soon available) scalar processors with HBM




317
311.9
306.7
301.6
296.5
291.3
286.2
281
275.9
270.8
265.6