Alexandru Calotoiu

From Control-Centric Programming to Data-Centric Optimization

Fortran with CLOUDSC

```
 9
10   SUBROUTINE CLOUDSC &
11    !---input
12    & (KIDIA,    KFDIA,    KLON,    KLEV, &
13    & PTSPHY,&
14    & PT, PQ, tendency_cml,tendency_tmp,tendency_loc, &
15    & PVFA, PVFL, PVFI, PDYNA, PDYNL, PDYNI, &
16    & PHRSW,    PHRLW,&
17    & PVERVEL,  PAP,      PAPH,&
18    & PLSM,     LDCUM,    KTYPE, &
19    & PLU,      PLUDE,    PSNDE,    PMFU,     PMFD,&
20    !---prognostic fields
21    & PA,&
22    & PCLV,  &
23    & PSUPSAT,&
24   !-- arrays for aerosol-cloud interactions
25   !!! & PQAER,    KAER, &
26    & PLCRIT_AER,PICRIT_AER,&
27    & PRE_ICE,&
28    & PCCN,     PNICE,&
29    !---diagnostic output
30    & PCOVPTOT, PRAINFRAC_TOPRFZ,&
31    !---resulting fluxes
32    & PFSQLF,   PFSQIF ,  PFCQNNG,  PFCQLNG,&
33    & PFSQRF,   PFSQSF ,  PFCQRNG,  PFCQSNG,&
34    & PFSQLTUR, PFSQITUR , &
35    & PFPLSL,   PFPLSN,   PFHPSL,   PFHPSN, KFLDX, &
36    & YDCST, YDTHF, YDECLDP)
```

- **Cloud Microphysics of IFS**
  - Resolve sub-grid features
  - Original 2,525 SLOC of Fortran 95
- **Rewritten for performance portability benchmarking (optimization took months!)**
  - 2,635 SLOC C
  - 2,610 SLOC C++/CUDA

https://github.com/ecmwf-ifs/dwarf-p-cloudsc

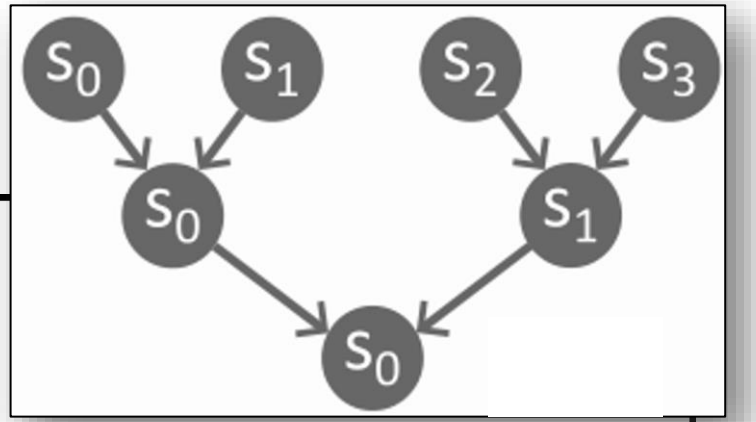
A first simple loop from CLOUDSC*

Data Parallelism	✓ ..... do JK=1,KLEV ✓ ..... do JL=1,KFDIA ZQSM(JL,JK)=ZQSM(JL,JK)/(1.0-RE*ZQSM(JL,JK)) enddo enddo
Work	KLEV * KFDIA
Depth	**1**
Average Parallelism	KLEV * KFDIA

Fully data parallel

* examples are simplified for presentation purposes

A second more complex loop from CLOUDSC



Data Parallelism	✗ ..... do JN=1, NSTEP-1 ✓ ..... do JL=1, KFDIA ZQXN(JL, NSTEP) = ZQXN(JL, NSTEP)+ZQXN(JL, JN) enddo enddo	(array) accumulation prevents parallelization ☹
Work	(NSTEP-1) * KFDIA	(NSTEP-1) * KFDIA
Depth	(NSTEP-1) * KFDIA	$\log_2$(NSTEP-1)
Average Parallelism	1	(NSTEP-1) * KFDIA / $\log_2$(NSTEP-1)

Now multiple realistic CLOUDSC loops

Data Parallelism

L1

```
X ..... do JM=1,4
X ..... do JK=1,KLEV
X ..... do JL=1,KFDIA
1   if ZQX(JL,JK,JM)<RLMIN) then
2     ZQADJ=ZQX(JL,JK,JM)*ZQTMST
3     tend_q(JL,JK)=tend_q(JL,JK)+ZQADJ
4     tend_T(JL,JK)=tend_T(JL,JK)-RAL*ZQADJ
5     ZQX(JL,JK,JM)=0.0
```

reuse of temporary variable prevents parallelization

L2

```
✓ ..... do JK=1,KLEV
✓ ..... do JL=1,KFDIA
6     ZQSM(JL,JK)=ZQSM(JL,JK)/(1.0-RE*ZQSM(JL,JK))
```

L3

```
✓ ..... do JK=1,KLEV
✓ ..... do JL=1,KFDIA
7     ZA(JL,JK)=MAX(0.0,MIN(1.0,ZA(JL,JK)))
8     ZLI(JL,JK)=ZQX(JL,JK,1)+ZQX(JL,JK,2)
9     if (ZLI(JL,JK)>RLMIN) then
10      ZLFRAC(JL,JK)=ZQX(JL,JK,1)/ZLI(JL,JK)
      else
11      ZLFRAC(JL,JK)=0.0
```

Order Constraints

Happens-before (→)

- 1 → 2 Control
- 2 → 3 RAW
- 2 → 4 RAW
- 2 → 5 WAR

No order constraint: L2 || L1

- 5 → 8 RAW
- 8 → 9 RAW
- 9 → 10 Control
- 9 → 11 Control

L3 → L1

	L1	L2	L3	L1	L2	L3
Work	4 * KLEV * KFDIA * (1+4)	KLEV * KFDIA	KLEV * KFDIA * 4	KLEV * KFDIA * 25		
Depth	log2(4) * 1 *1 * (1+2)	1	1 * 1 * (2+1)	8		
Average Parallelism	KLEV * KFDIA * 10/3	KLEV * KFDIA	KLEV * KFDIA * 4/3	KLEV * KFDIA * 25/8		

Transformations in DaCe – Performance Metaprogramming!

Expose parallelism

- Data management transformations:
  - Changing data container lifetime
  - Versioning data containers

1

- Loop Parallelization

2

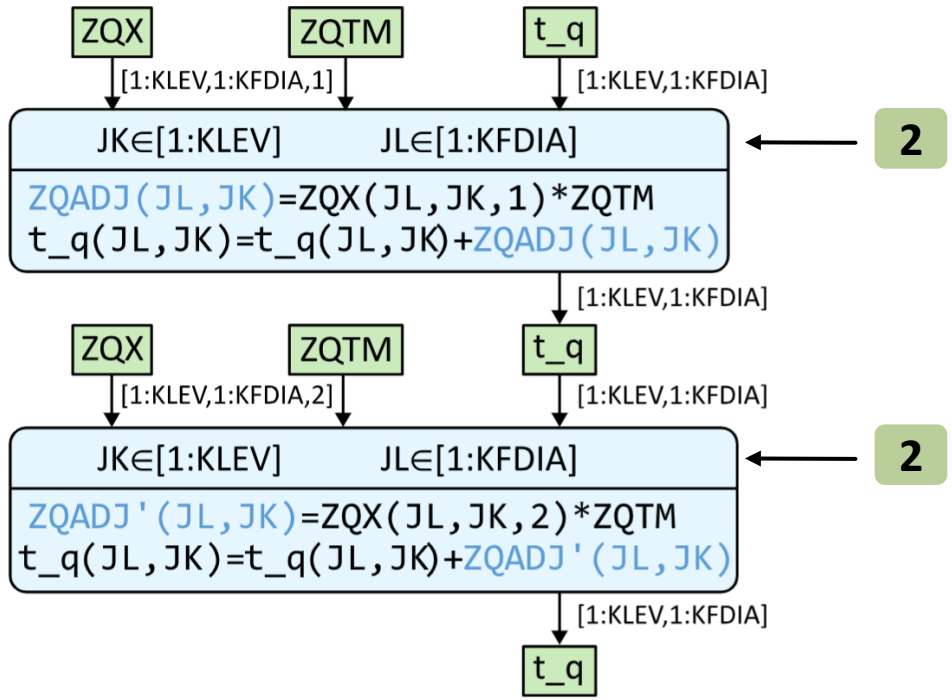
```
do JK=1,KLEV
  do JL=1,KFDIA
    ZQADJ=ZQX(JL,JK,1)*ZQTM
    t_q(JL,JK)=t_q(JL,JK)+ZQADJ
  enddo
enddo

do JK=1,KLEV
  do JL=1,KFDIA
    ZQADJ=ZQX(JL,JK,2)*ZQTM
    t_q(JL,JK)=t_q(JL,JK)+ZQADJ
  enddo
enddo
```

1

1

Improve performance

- Specializing numerical values
- Changing the data layout

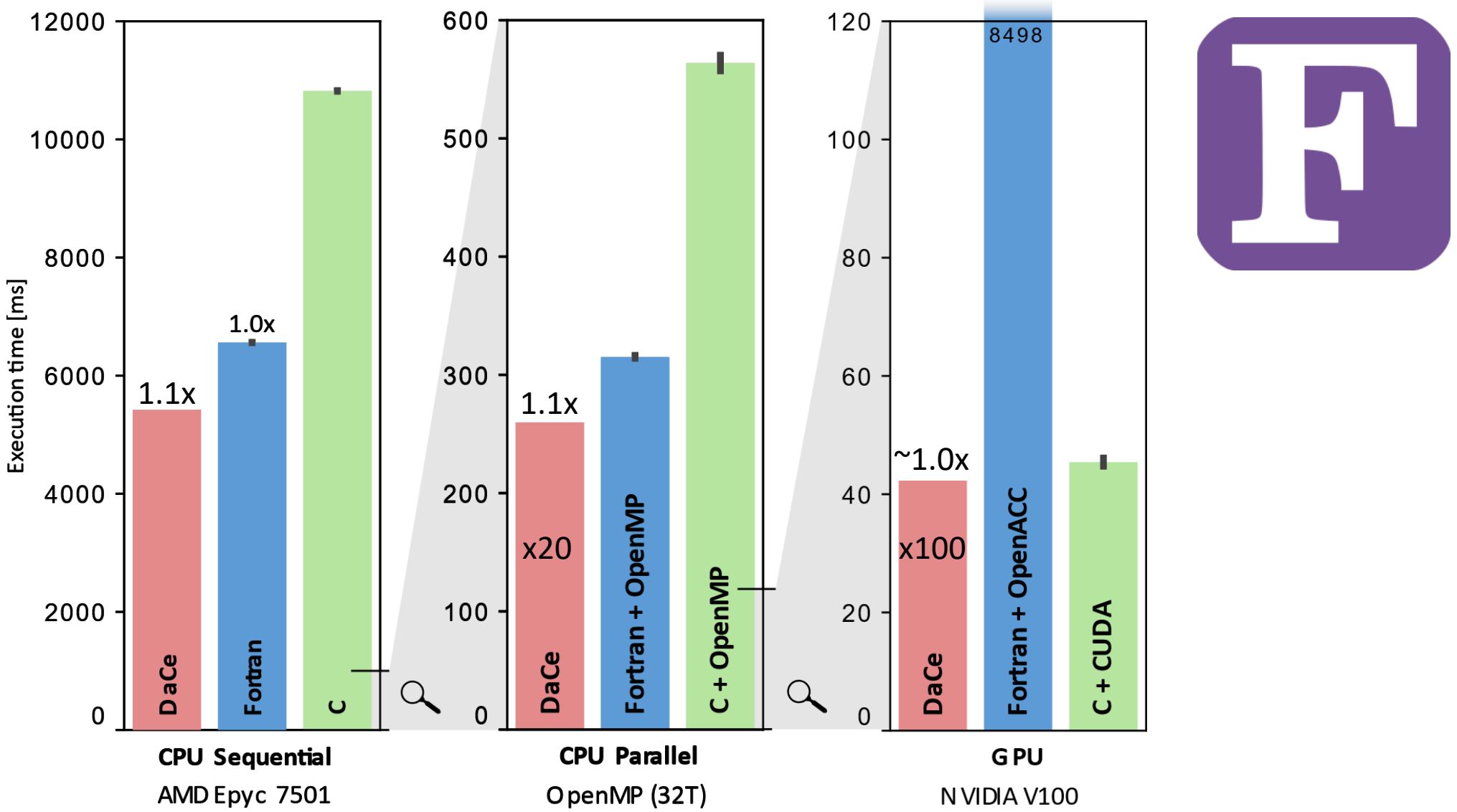


Minimizing depth!

```
! Function Definitions
REAL :: FOEEWM, FOELDCPM
FOEEWM ( PTARE ) = R2ES *&
   &(MIN(1.0,((MAX(RTICE,MIN(RTWAT,PTARE))-RTICE)*RTWAT_RTICE_R)**2)&
   & *EXP(R3LES*(PTARE-RTT)/(PTARE-R4LES))&
   & +(1.0-MIN(1.0,((MAX(RTICE,MIN(RTWAT,PTARE))-RTICE)*RTWAT_RTICE_R)**2))&
   & *EXP(R3IES*(PTARE-RTT)/(PTARE-R4IES)))
FOELDCPM ( PTARE ) = MIN(1.0,((MAX(RTICE,MIN(RTWAT,PTARE))-&
   &RTICE)*RTWAT_RTICE_R)**2)*RALVDCP+&
   &(1.0-MIN(1.0,((MAX(RTICE,MIN(RTWAT,PTARE))-RTICE)*RTWAT_RTICE_R)**2))*RALSDCP

! Vertical loop
DO JK=NCLDTOP,KLEV
...
    ! Loop Nest
    DO JL=KIDIA,KFDIA
        ZQP    = 1.0/PAP(JL,JK)
        ZQSAT = FOEEWM(ZTP1(JL,JK))*ZQP
        ZQSAT = MIN(0.5,ZQSAT)
        ZCOR   = 1.0/(1.0-RETV  *ZQSAT)
        ZQSAT = ZQSAT*ZCOR
        ZCOND = (ZQSMIX(JL,JK)-ZQSAT)/(1.0+ZQSAT*ZCOR*FOEDEM(ZTP1(JL,JK)))
        ZTP1(JL,JK) = ZTP1(JL,JK)+FOELDCPM(ZTP1(JL,JK))*ZCOND
        ZQSMIX(JL,JK) = ZQSMIX(JL,JK)-ZCOND
        ZQSAT = FOEEWM(ZTP1(JL,JK))*ZQP
        ZQSAT = MIN(0.5,ZQSAT)
        ZCOR   = 1.0/(1.0-RETV  *ZQSAT)
        ZQSAT = ZQSAT*ZCOR
        ZCOND1= (ZQSMIX(JL,JK)-ZQSAT)/(1.0+ZQSAT*ZCOR*FOEDEM(ZTP1(JL,JK)))
        ZTP1(JL,JK) = ZTP1(JL,JK)+FOELDCPM(ZTP1(JL,JK))*ZCOND1
        ZQSMIX(JL,JK) = ZQSMIX(JL,JK)-ZCOND1
    ENDDO
...
ENDDO
```

```
...

! Vertical loop
DO JK=NCLDTOP,KLEV
...
    ! Loop Nest
    DO JL=KIDIA,KFDIA
       ZQP_0(JL)    = 1.0/PAP(JL,JK)
    ENDDO
    DO JL=KIDIA,KFDIA
       ZQSAT = FOEEWM(ZTP1(JL,JK))*ZQP_0(JL)
       ZQSAT = MIN(0.5,ZQSAT)
       ZCOR   = 1.0/(1.0-RETV  *ZQSAT)
       ZQSAT = ZQSAT*ZCOR
       ZCOND_0(JL) = (ZQSMIX(JL,JK)-ZQSAT)/(1.0+ZQSAT*ZCOR*FOEDEM(ZTP1(JL,JK)))
    ENDDO
    DO JL=KIDIA,KFDIA
       ZTP1(JL,JK) = ZTP1(JL,JK)+FOELDCPM(ZTP1(JL,JK))*ZCOND_0(JL)
    ENDDO
    DO JL=KIDIA,KFDIA
       ZQSMIX(JL,JK) = ZQSMIX(JL,JK)-ZCOND_0(JL)
    ENDDO
    DO JL=KIDIA,KFDIA
       ZQSAT = FOEEWM(ZTP1(JL,JK))*ZQP_0(JL)
       ZQSAT = MIN(0.5,ZQSAT)
       ZCOR   = 1.0/(1.0-RETV  *ZQSAT)
       ZQSAT = ZQSAT_4(JL)*ZCOR
       ZCOND1_0(JL)= (ZQSMIX(JL,JK)-ZQSAT)/(1.0+ZQSAT*ZCOR*FOEDEM(ZTP1(JL,JK)))
    ENDDO
    DO JL=KIDIA,KFDIA
       ZTP1(JL,JK) = ZTP1(JL,JK)+FOELDCPM(ZTP1(JL,JK))*ZCOND1_0(JL)
    ENDDO
    DO JL=KIDIA,KFDIA
       ZQSMIX(JL,JK) = ZQSMIX(JL,JK)-ZCOND1_0(JL)
    ENDDO
...
ENDDO
```

Performance portability – all from the original, unchanged CLOUDSC code!



Conclusion

Canonicalization is key to performance & portability