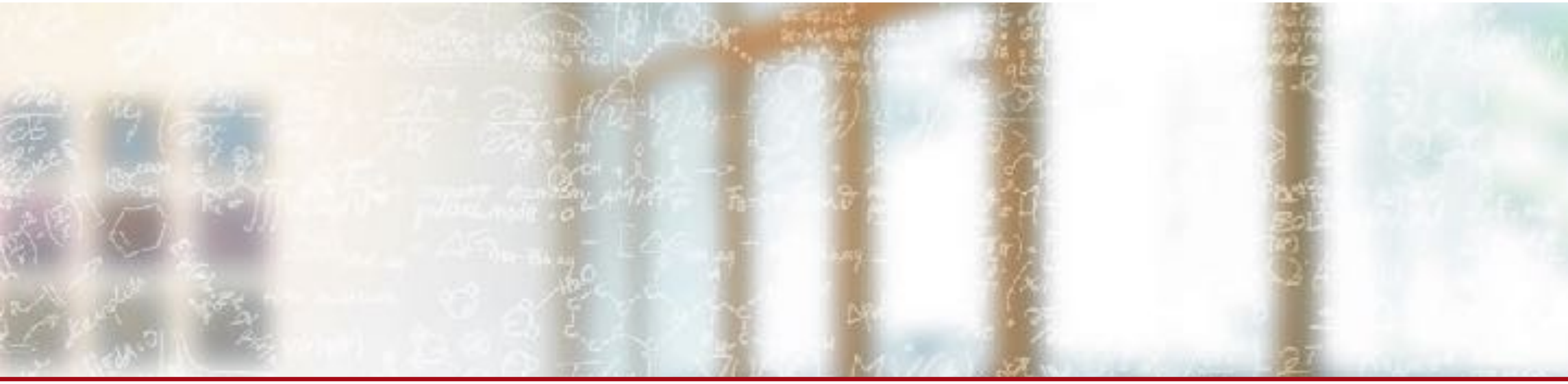
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Evolving Fortran for Emerging Architectures: Lessons from the ICON-GPU Atmospheric Model

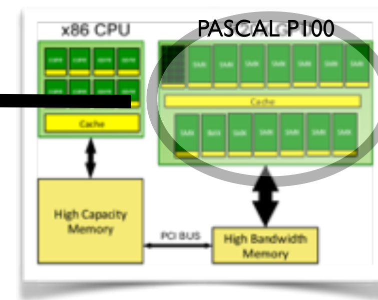
William Sawyer (CSCS), X. Lapillonne (MCH), D. Alexeev (NVIDIA), R. Dietlicher (MCH), M. Giorgetta (MPI-M), L. Kornblueh (MPI-M), S. Rast (MPI-M), R. Schnur (MPI-M), M. Esch (MPI-M), U. Schulzweida (MPI-M), Jan-Frederick Engels (DKRZ), M. Hanke (DKRZ), V. Clement (ORNL), P. Marti (ETH), R. Pincus (CU Boulder), G. Zängl (DWD), and others at **MPI-M, DKRZ, DWD, KIT, ETH** and in the **OpenACC** Organization

International Fortran Conference 2020
July 2, 2020, Zurich, Switzerland

Brief Introduction to CSCS

CSCS has numerous customers from several scientific communities:

- Computational Chemistry and Material Science
- Climate / Numerical Weather Prediction
- Seismology, Solid-Earth dynamics
- Life sciences
- others…



www.top500.org:
#10 in the world

green500:
#26 energy-efficiency

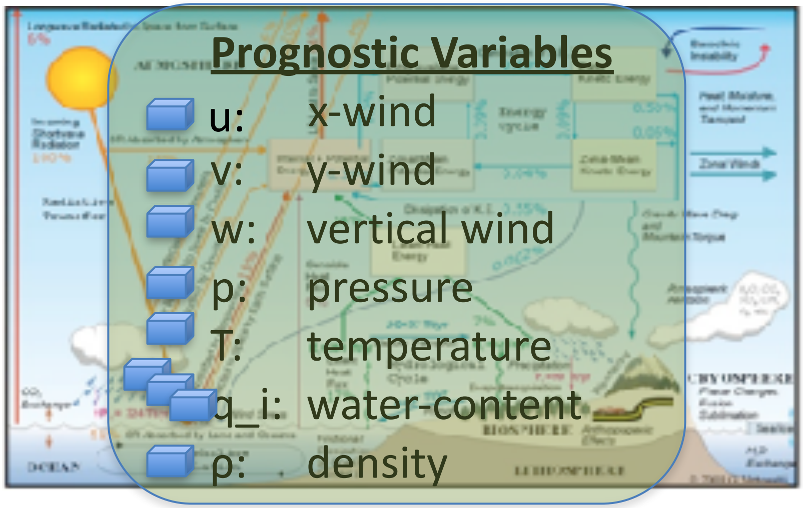
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Outline

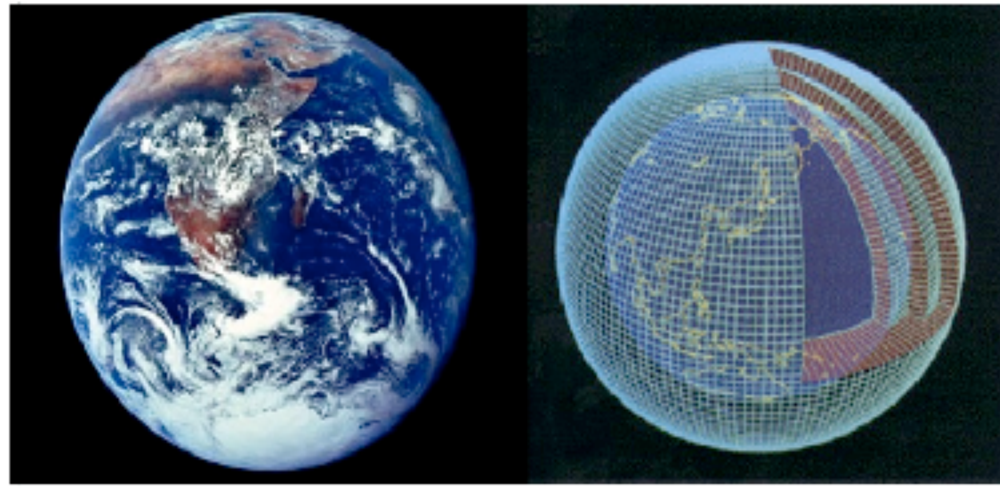
- One-page introduction to atmospheric modeling
- OpenCL and CUDAFortran prototype implementations of atmospheric dynamics solver ("dycore")
- OpenACC production implementation, results
- Lessons learned: positives and negatives
- From the user standpoint: How could Fortran evolve to address the negatives?
- Pointer: *Highly Parallel Fortran and OpenACC Directives*, Jeff Larkin, Friday (3.7) at 20:20 (CEDT)

One-slide introduction to atmospheric modeling

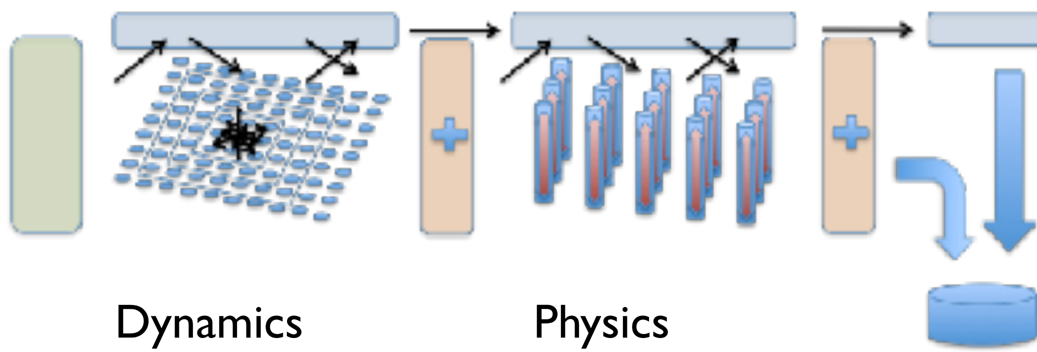


Prognostic Variables

- u: x-wind
- v: y-wind
- w: vertical wind
- p: pressure
- T: temperature
- q_i: water-content
- ρ: density



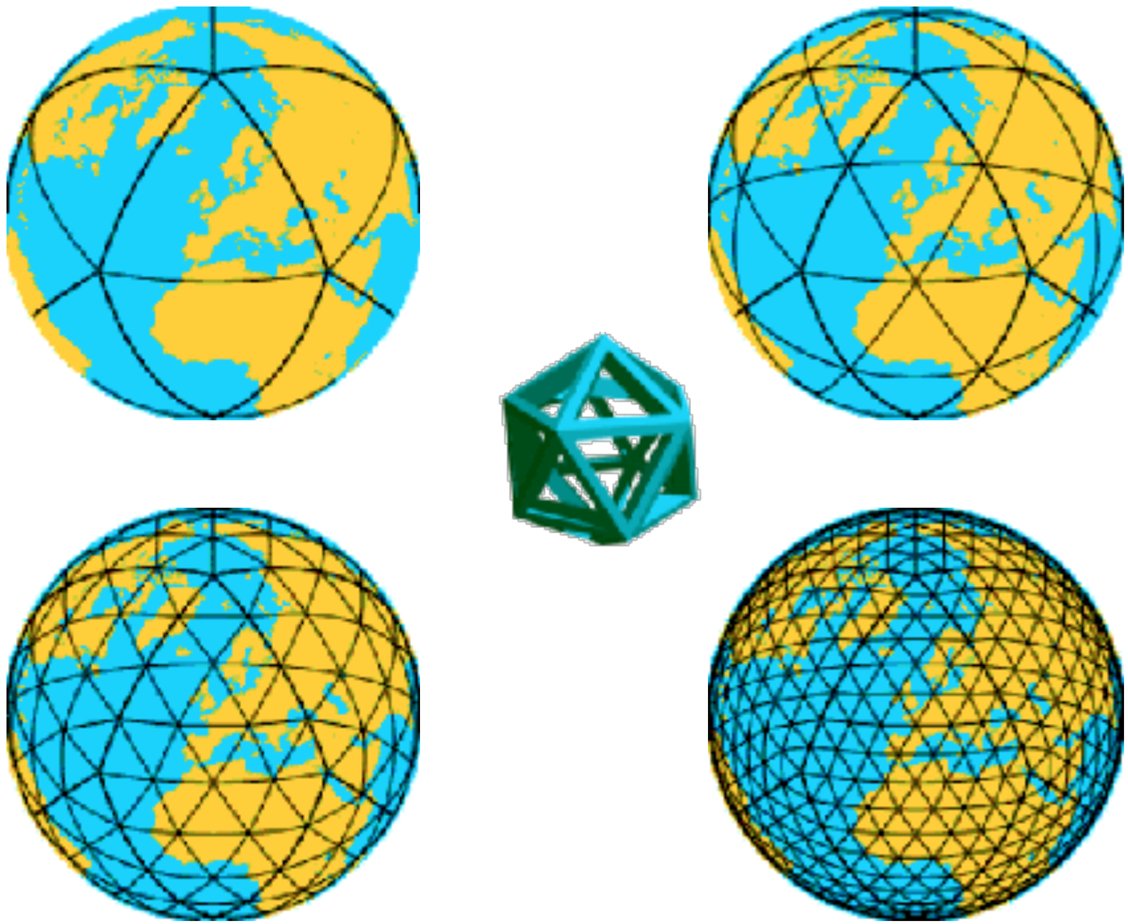
Dynamics: solve the 3-D equations of motion on rotating sphere
Physics: parameterize sub-grid phenomena on vertical profiles,
➔ turbulence, hydrological processes, radiation, gravity wave drag...



ICOsahedral Non-hydrostatic modeling framework

- Next generation global NWP and climate modeling system, successor of European Centre Hamburg Model (ECHAM)
- Joint development German Weather Service (DWD) and Max Planck Institute for Meteorology (MPI-M), German Climate Computing Center (DKRZ) and Karlsruhe Institute of Technology (KIT)
- Non-hydrostatic atmospheric dynamics solver ("dycore") on icosahedral-triangular grid, coupled with previously mentioned physical parameterizations
- Roughly 1M lines of Fortran-2003 code, with some utilities in C++
- Under development: Hydrostatic ocean model using (basically) same grid structure

ICON Horizontal Grid

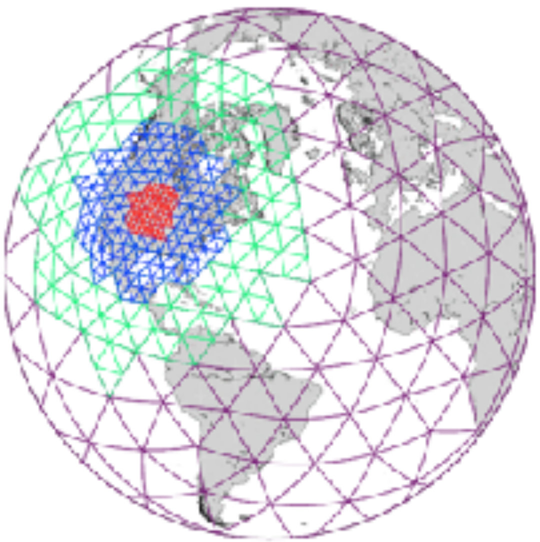


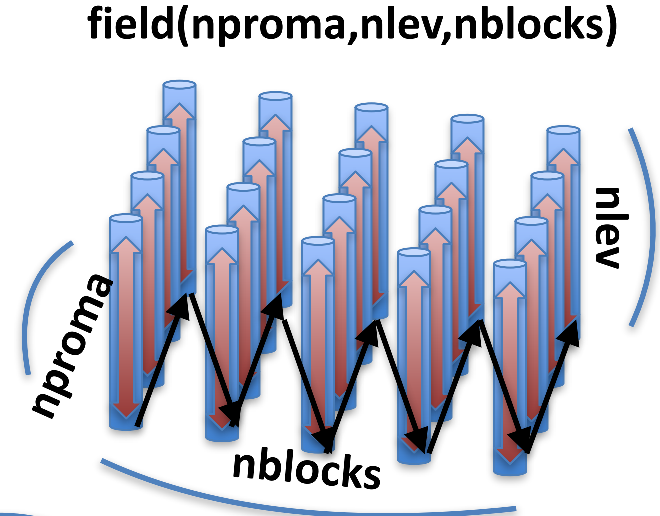
R2B0

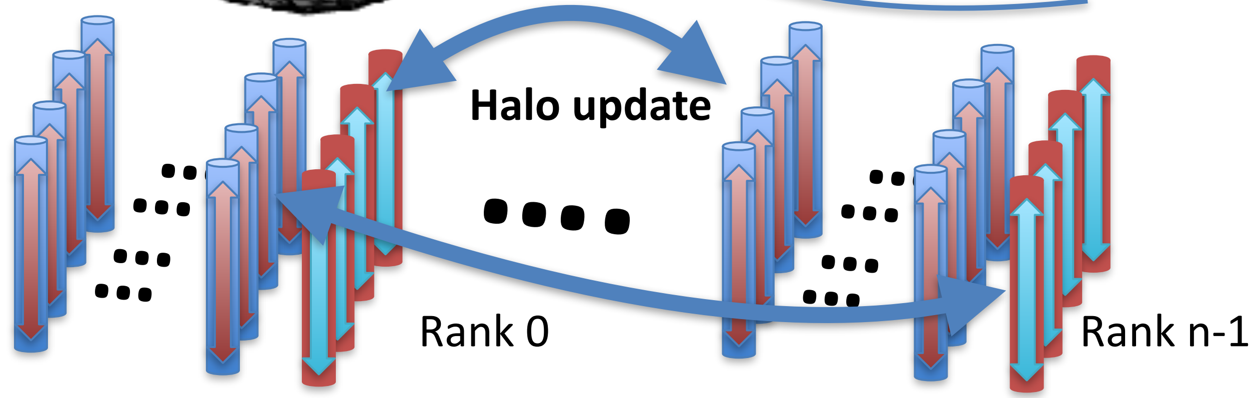
R2B1

R2B2

ICON model: grid, data management and domain decomposition







Credits: ICON team, Behrens, Fukuda, et al.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Prototype solvers on GPUs (2012)

Fortran

```
DO jb = i_startblk, i_endblk

  CALL get_indices_c(p_patch, jb, i_startblk, i_endblk, &
                     i_startidx, i_endidx, rl_start, rl_end)

  DO jk = 1, nlev
    DO jc = i_startidx, i_endidx
```

OpenCL

```
const int jb = i_startblk + get_global_id(0);
const int jc = localStart[get_global_id(0)] + get_global_id(2);
const int jk = get_global_id(1);

if (jk < nlev && jb < i_endblk && jc < localEnd[get_global_id(0)])
{
    const int idx = jc + jk*nproma + jb*nproma*nlev;
```

CUDAFortran

```
jb = blockidx%x + ( i_startblk -1 )
je = threadidx%x
jk = threadidx%y

IF ( ( i_startblk < jb .and. jb < i_endblk ) .or. &
     ( i_startblk == jb .and. i_startidx <= je ) .or. &
     ( i_endblk == jb .and. je <= i_endidx ) ) THEN
```

OpenACC ICON GPU porting steps

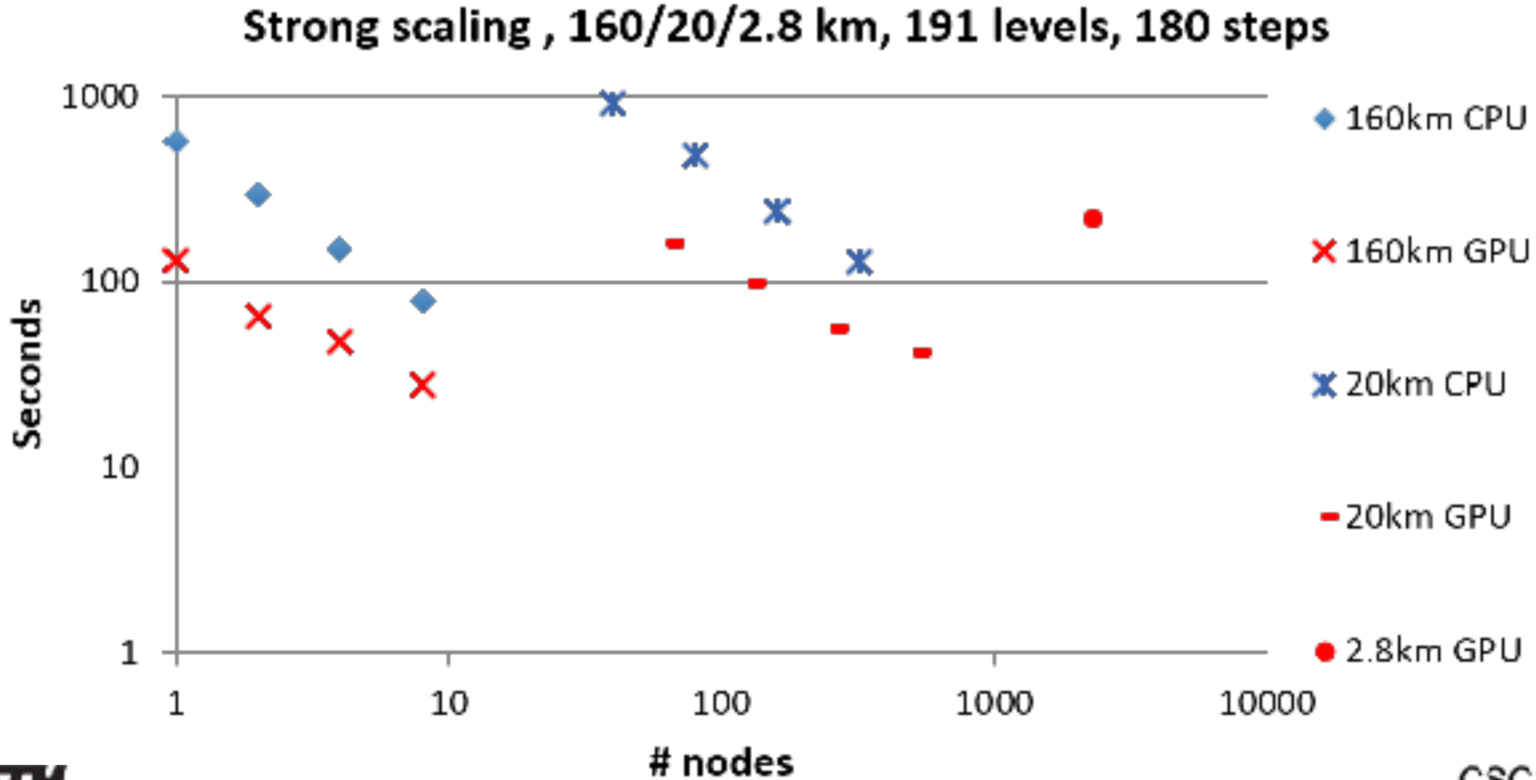
- Dycore: PRACE 2IP-funded project 2012-15, ENIAC 2017-2019
  - OpenACC directives, challenges with derived data types
  - Platform for Advanced Scientific Computing ENIAC project: Physics-compatible data layout (very large nproma)
- Physics: implemented within PASC ENIAC project (2017-2019)
  - *Microphysics: OpenACC directives*
  - *Turbulence ("vdiff"): OpenACC directives*
  - *Land-surface ("JSBACH"): CLAW preprocessor + OpenACC*
  - *Radiation: "RRTMGP": OpenACC*
  - *Saturation adjustment: OpenACC*
  - *Use of NVIDIA CUB-library to avoid OpenACC Atomics*

ICON scaling (PASC Review, 2020)

"CPU": node with 1x Haswell sockets

"GPU": nodes with 1x P100

Bottom line: speedup ~4.9x



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Lessons learned 2010-2017

- New technologies (OpenCL, domain-specific languages) or proprietary tools (CUDAFortran) very difficult for this community
- Directives (e.g., OpenACC) acceptable, but have major impact on code readability and maintainability
- ICON is monolithic, does not have unit and component tests (!)
  Creating validation infrastructure was much bigger job than GPU port!
- Physical parameterizations are 0-D (box) or 1-D column models, *but developers optimize code for 3-D; no separation of concerns*

Why are these experiences important for Fortran?

- Climate and NWP software developers are among the staunchest Fortran supporters
- New models are being written using advanced Fortran functionality (F08 and above) as we speak, e.g., UKMetOffice LFRic
- Longevity of "open" directive standards is not assured; witness competition OpenMP-acc vs. OpenACC
- Directives are "band-aids" for missing compiler functionality: the combination of directives for various target architectures is a mess

Directives and CPP case distinctions diminish maintainability

```
!$OMP PARALLEL PRIVATE (rl_start,rl_end,i_startblk,i_endblk)
      rl_start = 5
      rl_end   = min_rledge_int - 2
      i_startblk = p_patch%edges%start_block(rl_start)
      i_endblk   = p_patch%edges%end_block(rl_end)
#ifdef __SX__
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,jk,je,z_vn_avg,zaux) ICON_OMP_DEFAULT_SCHEDULE
#else
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,jk,je,z_vn_avg) ICON_OMP_DEFAULT_SCHEDULE
#endif
      DO jb = i_startblk, i_endblk
        CALL get_indices_e(p_patch, jb, i_startblk, i_endblk, &
                           i_startidx, i_endidx, rl_start, rl_end)
        IF (istep == 1) THEN
!$ACC PARALLEL IF( i_am_accel_node .AND. acc_on )  DEFAULT(NONE) ASYNC(1)
          !$ACC LOOP GANG VECTOR COLLAPSE(2)
#ifdef __LOOP_EXCHANGE
          DO je = i_startidx, i_endidx
!DIR$ IVDEP
            DO jk = 1, nlev
#else
!$NEC outerloop_unroll(8)
          DO jk = 1, nlev
!$NEC vovertake
            DO je = i_startidx, i_endidx
#endif
#ifndef __SX__
   :
```

Addition of OpenMP 4.5 accelerator directives would require more #ifdefs !

Anatomy of "performance portable" code

```
!$OMP PARALLEL PRIVATE (rl_start,rl_end,i_startblk,i_endblk)
      rl_start = 5
      rl_end    = min_rledge_int - 2
      i_startblk = p_patch%edges%start_block(rl_start)
      i_endblk    = p_patch%edges%end_block(rl_end)
#ifdef __SX__
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,jk,je,z_vn_avg,zaux) ICON_OMP_DEFAULT_SCHEDULE
#else
!$OMP DO PRIVATE(jb,i_startidx,i_endidx,jk,je,z_vn_avg) ICON_OMP_DEFAULT_SCHEDULE
#endif
      DO jb = i_startblk, i_endblk
        CALL get_indices_e(p_patch, jb, i_startblk, i_endblk, &
                           i_startidx, i_endidx, rl_start, rl_end)
        IF (istep == 1) THEN
!$ACC PARALLEL IF( i_am_accel_node .AND. acc_on ) DEFAULT(NONE) ASYNC(1)
          !$ACC LOOP GANG VECTOR COLLAPSE(2)
#ifdef __LOOP_EXCHANGE
          DO je = i_startidx, i_endidx
!DIR$ IVDEP
            DO jk = 1, nlev
#else
!$NEC outerloop_unroll(8)
          DO jk = 1, nlev
!$NEC vovertake
            DO je = i_startidx, i_endidx
#endif
#ifndef __SX__
          :
```

Multithreading or 'multistream' DO loop at high level

DO CONCURRENT tries to address both of these

SIMD, 'vector', or dependency-free loops compiler needs to 'hoist' loops if needed

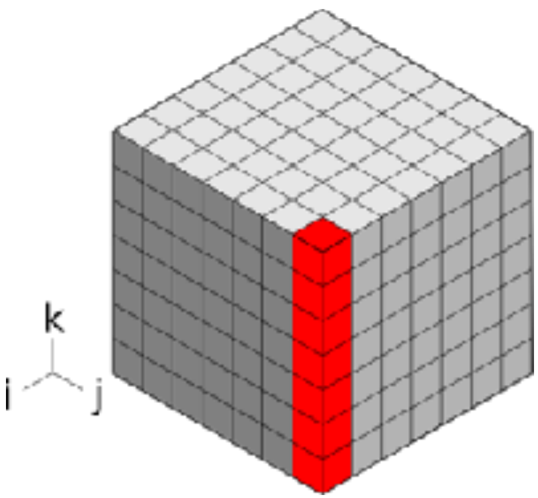
Data must be "cached" in accelerator memory so that all computations take place without host-device

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CLAW: Single column abstraction for Physics

Dependency on the vertical dimension only: *scientist prefer to develop column models and ignore parallelization issues*

Algorithm for one column only



CLAW is a Fortran source-to-source translator which generates Fortran + OpenMP or OpenACC

```
SUBROUTINE rhs_bksub(...)
 !Declarations

 !$claw define dimension kl(1:nproma) &
 !$claw parallelize
 DO jvar = 1,nvar_vdiff
   im = matrix_idx(jvar)
   DO jk = ibtm_var(jvar)-1,itop,-1
      jkp1 = jk + 1
      bb(jk,jvar) =  bb(jk   ,jvar) &
                   & -bb(jkp1,jvar) &
                   & *aa(jk   ,3,im)

   ENDDO
 ENDDO
END SUBROUTINE rhs_bksub
```

https://github.com/claw-project/claw-compiler

ICON Take-home messages

- Climate modeling community: devoted Fortran developers, not very adaptable to new technologies
- GPUs and distinct memory spaces are here to stay
- ICON (climate) now successfully implemented on Nvidia GPUs using OpenACC and (few) calls to the Nvidia CUB library
- Key problem in port: managing the data on device — can be considered a software-managed cache
- Directives are 'band-aids' for missing compiler functionality: to support such architectures, Fortran needs to evolve
- DO CONCURRENT tries to cover both multi-threaded and SIMD parallelism. Will it be successful?
- *Highly Parallel Fortran and OpenACC Directives*, Jeff Larkin, Friday (3.7) at 20:20 (CEDT)

Thanks for listening!

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich