The Good, the Ugly and the Bad:
What We Learned from Porting ICON to GPUs


OpenACC Summit 2021
Sep. 14, 2021, virtual
Approaches to parallel programming from user perspective

From discussions with Jeff Larkin

```
Y += A*X
```

**Domain Specific Language**

**Base language support**

```
do concurrent (i = 1: n)
   y(i) = a*x(i)+y(i)
enddo
```

**Directives / pragmas**

```
!$acc parallel loop
do j = 1,n
   y(j) = y(j) + a * x(j)
enddo
!$acc end parallel loop
```

**Language extensions / Intrinsics**

```
attributes(global) subroutine daxpy(n,a,x,y)
double precision, dimension(*) :: x,y
double precision, value :: a
integer, value :: n, i
i = (blockidx%x-1) * blockdim%x + threadIdx%x
if( i <= n ) y(i) = a * x(i) + y(i)
end subroutine
```
ICON in the nutshell

• ICON is ecosystem of atmospheric and ocean modeling software enabling climate and numerical weather prediction

• Developed by ~200 people, 4 German member institutions + numerous others, about 2M lines of code written from 2001 to today

• Successor of COSMO (regional atmospheric Climate/NWP model):
  • ICON for forecasting: DWD in 2015, MeteoSwiss in 2022
  • Also ICON for numerous climate simulations, transition in progress
One-slide introduction to (atmospheric) modeling

**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...
Timeline: ICON GPU port
Large CSCS investment

2010-2019: Port of COSMO with DSL (dynamics) and OpenACC (physics)
2011: ICON dycore (solves atmos. eqns) prototypes (CUDAFortran / OpenCL)
  → ICON developers insist on directive-based approach
2013-2016: PRACE 2IP Work Package 8: ICON dynamical core, one of ~15 applications chosen for HPC refactoring, based on OpenACC directives
2015-2017: Effort to port physics of ICON-HDCP^2 to GPUs unsuccessful: scientific development too fast, no component testing infrastructure
2015-2019: (Pincus, Norman et al.) OpenACC port RRTMGP radiation: advice
2017-2020: PASC ENIAC project to port climate-physics, partially with new tools
2017-2018: ENIAC port of PSrad physics unsuccessful, reverted to RRTMGP
2018-2019: dynamical core refactored to match physics data layout
2019: “final push” GPU-programming ‘hackathon’, intensive effort to incorporate RRTMGP, additional optimizations, extensive testing, system integration
2020-2021: QBO simulations in production at CSCS (support effort)

Enabling ICON for Kilometer-Scale Global Climate on GPU Systems; Sawyer, William, PASC19 MS08 - “Bridging the Software Productivity Gap for Weather and Climate Models, Part II of II”
ICON Horizontal Grid

R2B0

R2B1

R2B2
ICON the Good: OpenACC

- **It works with good performance after careful optimization**
- Good initial support from Cray for CCE compiler
- Subsequent vendor support from PGI/Nvidia
  - 2013 - 2017: PGI cannot compile ICON for CPUs; *Dave Norton tracks down and reports ~20 compiler bugs.* PGI 18.x works
  - 2019: OpenACC Atomics in index list generation too slow. *Dmitry Alexeev replaces atomics with calls to CUB library*
  - 2019-21: *Dmitry introduces ASYNC and other OpenACC optimizations* (e.g., A100), in particular in RRTMGP radiation
- CSCS has strong bonds to OpenACC community
  - Participate in weekly technical calls (user perspectives)
  - Thomas Schulthess elected board member (2019)
- GPU port for climate simulations (QUBICCC) ultimately successful
  - Roughly 5x speedup on P100 w.r.t., single-socket Haswell
  - Port from CSCS Daint (P100) to JSC Juwels Booster (A100) straightforward
Intel Haswell, AMD EPYC, Nvidia P100/V100/A100 Performance

Single-node performance (R2B04 = 160km)

Benchmarking: Dmitry Alexeev
Intel Haswell, AMD EPYC, Nvidia P100/V100/A100 Performance

Strong-scaling (R2B07 == 20km)

Benchmarking: Dmitry Alexeev
End-to-end benchmarks (QUBICCC proposal)

**CPU1**: nodes 1xHaswell

**CPU2**: nodes 2xBroadwell

**GPU1**: P100
communication G->C->C->G

**GPU2**: P100
communication G->G
ICON the Ugly

- ICON is a monolithic code; no unit/component tests (or lost after initial development). Similar to COSMO in this respect
  - Testing infrastructure needed for GPU development (months)
- Original PRACE 2IP dynamical core parallelization not designed with entire model in mind
  - Dycore required refactoring during port of full model (weeks)
- PASC funding for GPU port time-limited
  - ENIAC delayed, team barely completed port of climate “Physics”
  - Component integration into full model by CSCS
Ugly: Dynamics refactoring needed for large block sizes

Original OpenMP code

SUBROUTINE solve_nonhydrostatic_eqns

!$OMP PARALLEL
    !$OMP DO PRIVATE( lots of vars )
    DO jb = 1, nblocks
        DO jk = 1, nlev
            DO jc = 1, nproma
                prog_var(jc,jk,jb) = f(jc,jk,jb)
            END DO
        END DO
    END DO
END DO
!$OMP END DO NOWAIT

END SUBROUTINE solve_nonhydrostatic_eqns

Original OpenACC

SUBROUTINE solve_nonhydrostatic_eqns

!$ACC PARALLEL
    !$ACC LOOP GANG
    !$ACC LOOP VECTOR COLLAPSE(2)
    DO jb = 1, nblocks
        DO jk = 1, nlev
            DO jc = 1, nproma
                prog_var(jc,jk,jb) = f(jc,jk,jb)
            END DO
        END DO
    END DO
END ACC PARALLEL

END SUBROUTINE solve_nonhydrostatic_eqns

New OpenACC

SUBROUTINE solve_nonhydrostatic_eqns

!$ACC PARALLEL
    !$ACC LOOP GANG
    DO jb = 1, nblocks
        !$ACC LOOP VECTOR COLLAPSE(2)
        DO jk = 1, nlev
            DO jc = 1, nproma
                prog_var(jc,jk,jb) = f(jc,jk,jb)
            END DO
        END DO
    END DO
END ACC PARALLEL

END SUBROUTINE solve_nonhydrostatic_eqns
Scientists are not Software Engineers

- ICON developers generally do not write unit tests
- New code features are directly incorporated into model, often with a namelist flag to toggle them
- But: refactoring the feature requires compilation of all of ICON (remember PGI compilation problems)
- For GPU porting: it is *much* easier to port code (e.g. physics) in standalone driver, with serialized data from real model run
- [https://github.com/GridTools/serialbox](https://github.com/GridTools/serialbox) (Arteaga, et al.) serialization, includes ppser.py to preprocess serialization directives
- [https://github.com/fortesg/fortrantestgenerator](https://github.com/fortesg/fortrantestgenerator) (Hovy) generating unit tests for subroutines of existing Fortran applications

Thanks for your attention
Ugly: Long-term support needed for tools

CLAW Compiler (Clement et al.)
- Source-to-source translator
- Based on the OMNI Compiler Project
- Fortran 2008
- Open source under the BSD license
- [https://github.com/claw-project/claw-compiler](https://github.com/claw-project/claw-compiler)
- Generation of OpenACC/OpenMP directives on the fly

CLAW Single-Column Abstraction (SCA)
- High-level abstraction for weather and climate code
- Targets physical parameterization: column or box models
- Achieve portability and performance portability

ICON the Bad: Changing messages on OpenACC

- OpenACC commitment based on Cray’s early enthusiasm (e.g., John Levesque’s presentations ~2012), requirements of community
- Luiz DeRose intimates shift from OpenACC to OpenMP (2015)
- 2019: Cray announces OpenACC unsupported in CCE 9.x
  - MeteoSwiss/CSCS already in multi-year transition to PGI (painful, but successful)
- 2020: For LUMI, HPE/Cray promises ‘sufficient’ CCE support to compile ICON benchmark code (but no more than that)
- July 2021: HPE commits to support OpenACC 3.x / OpenMP 5.x in “directive-agnostic” fashion
Bad: we used unofficial extensions

```c
#if defined( _OPENACC )
    CALL init_gpu_variables( )
    CALL save_convenience_pointers( )
!$ACC DATA COPYIN( p_int_state, p_patch, p_nh_state, prep_adv ), IF
( i_am_accel_node )
    CALL refresh_convenience_pointers( )
#endif
TIME_LOOP: DO jstep = (jstep0+jstep_shift+1), (jstep0+nsteps)
    :
ENDDO TIME_LOOP
#if defined( _OPENACC )
    CALL save_convenience_pointers( )
!$ACC END DATA
    CALL refresh_convenience_pointers( )
    CALL finalize_gpu_variables( )
#endif
```

Cray CCE full automated deep copy
Bad: Transition to manual deep copy

- Protracted discussion in OpenACC committee
- In 2018 we stopped waiting

```fortran
!$ACC ENTER DATA &
!$ACC COPYIN( p_int(j)%lsq_high, p_int(j)%lsq_lin, &
!$ACC              p_int(j)%c_bln_avg, p_int(j)%c_lin_e, p_int(j)%cells_aw_verts, &
!$ACC              p_int(j)%e_bln_c_s, p_int(j)%e_flx_avg, p_int(j)%geofac_div, &
!$ACC              p_int(j)%geofac_grdiv, p_int(j)%geofac_grg, p_int(j)%geofac_n2s, &
!$ACC              p_int(j)%geofac_rot, p_int(j)%lsq_high%lsq_blk_c, &
!$ACC              p_int(j)%lsq_high%lsq_dim_stencil, p_int(j)%lsq_high%lsq_idx_c, &
!$ACC              p_int(j)%lsq_high%lsq_moments, p_int(j)%lsq_high%lsq_moments_hat, &
!$ACC              p_int(j)%lsq_high%lsq_pseudoinv, p_int(j)%lsq_high%lsq_qmat_c, &
!$ACC              p_int(j)%lsq_high%lsq_rmat_utri_c, p_int(j)%lsq_high%lsq_weights_c &
: &
: &
```

Subsequently: all CCE-specific code removed
Take home messages

- **Good:**
  - OpenACC is a successful approach in the absence of base language support
  - Successful collaboration; CSCS integrated into development
  - We built good relationships with the OpenACC community
  - Great support from Nvidia; good performance (QUBICC)
  - Partners now see value in GPUs; allocation requests to follow
  - Follow-on efforts (EXCLAIM, ESiWACE) have good promise

- **Ugly:**
  - ICON is monolithic, component testing and porting difficult
  - OpenACC has serious deficiencies, but also other tools like CLAW
  - Personnel bottlenecks (short-term contracts), long-term support needed

- **Bad:**
  - Cray dropped OpenACC support in 2018; *HPE reverses strategy in July 2021*
  - Usage of non-standard features for automated deep copy
  - Bottlenecks / refactoring meant bigger CSCS effort than foreseen
  - CSCS lost the high-visibility PRACE QUBICC project to Jülich Juwels Booster

*Thanks for your attention*