Hewlett Packard
Enterprise

Porting ICON to AMD GPUs: Lessons Learned

Peter Wauligmann, Aniello Esposito

June 27, 2023

Introduction

- The **ICO**sahedral **N**on-hydrostatic (ICON) model is a key weather and climate application
- Developed by the German Weather Service (DWD) with a lot of partners (MPI-M, KIT, DKRZ, CSCS, …)

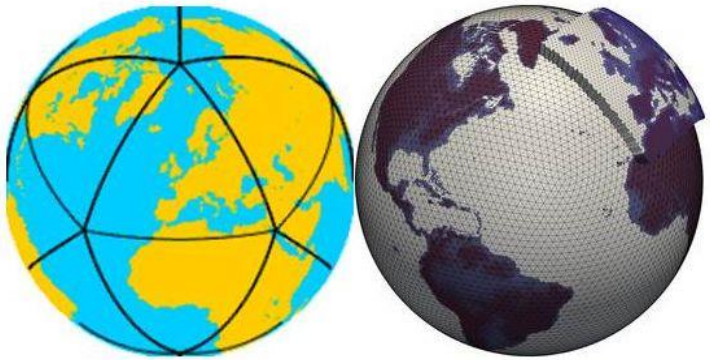


Illustration of spatial discretization in ICON using icosahedrons [1]

- ICON was an important part of the LUMI procurement
- Version: icon-2.6.0 (older than 3 years by now)
- Benchmark: 24 hours simulation using the global R02B10 grid (2.5 km resolution)
- Hardware: All available nodes of LUMI-G (AMD MI250X GPUs)

[1] https://www.dwd.de/EN/research/weatherforecasting/num_modelling/bilder/01_icon_modellgitter_en.jpg

ICON‘s programming models and how to port them

- Fortran is the main programming language
- Distributed memory parallelization using MPI
- OpenMP for CPU shared memory parallelization
- OpenACC is used to offload ~1700 kernels to the GPU
- CUDA libraries are used for parallel primitives (NVIDIA’s CUB)

Why was porting to AMD MI250X and LUMI a challenge?

1. CUDA sections had to be ported to HIP
2. ICON’s GPU version has been exclusive to NVIDIA accelerators so far
3. Missing compiler support for Fortran + OpenACC + AMD accelerators
4. HPE extended the Cray Compiling Environment (CCE) to cover the required OpenACC functionality
5. Early compiler features expectedly contain (performance) bugs
6. Running at scale on a new system

Hipification of CUDA sections

- ICON uses optimized NVIDIA library functions for primitives such as DeviceSelect from CUB
- AMD provides identical functionality in the hipCUB library

Functionality provided by DeviceSelect by the CUB library [2]

- Replace „cuda“ with „hip“ and all CUDA libraries with ROCM libraries
- Compile with hipcc instead of nvcc
- CCE prefers Fortran device pointer to be passed with TYPE(c_ptr) instead of TYPE(c_devptr)
- On NVIDIA hardware, CUB and OpenACC are performed on the same stream using acc_get_cuda_stream
- HPE compiler team extended CCE by acc_get_hip_stream function

[2] https://nvlabs.github.io/cub/select_logo.png

Bridging differences between the CCE and NVFORTRAN

- ICON used `acc_attach(ptr)` which was not part of the OpenACC specification for Fortran
  - Replaced `acc_attach(ptr)` by `!$ACC ENTER DATA ATTACH(ptr)`
  - The latest OpenACC specification from November 2022 introduced an interface for Fortran

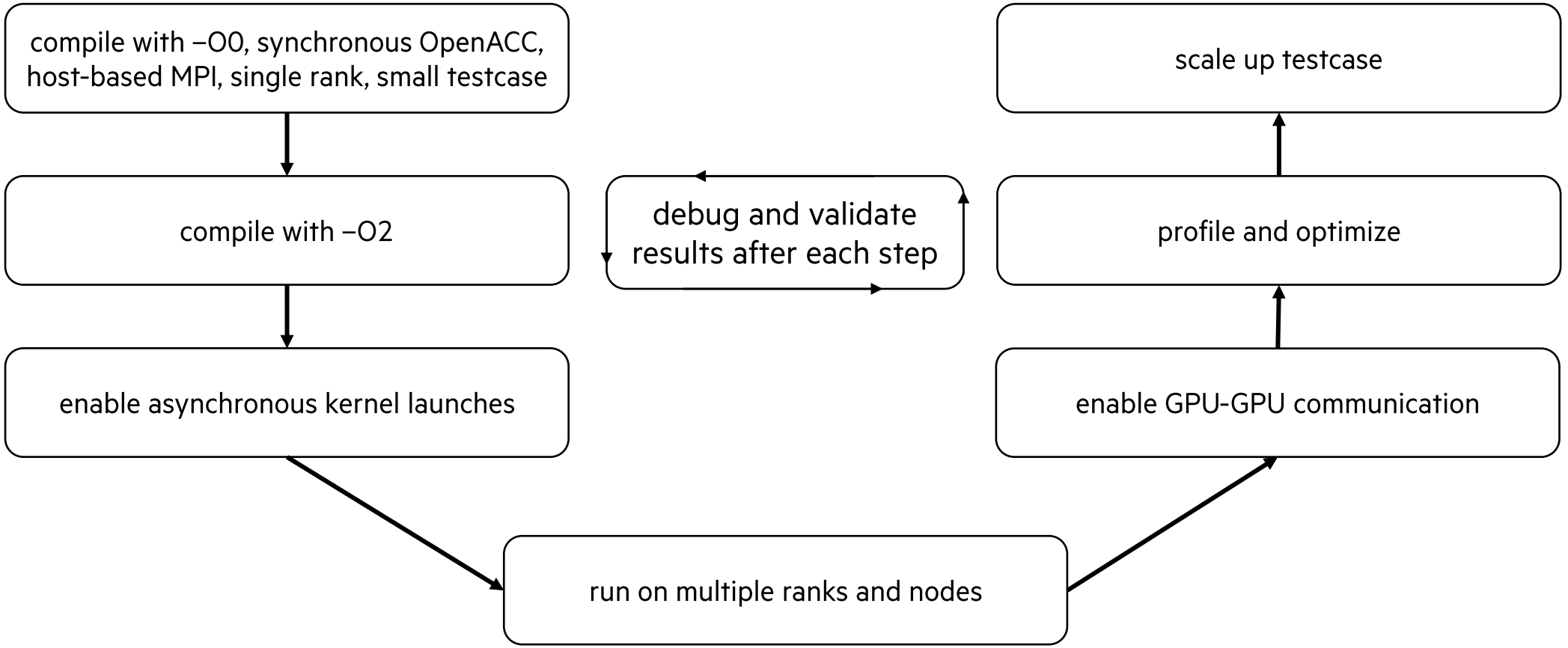
- NVIDIA and HPE implement the `DEFAULT(NONE)` clause differently
  - NVFORTRAN requires only the data attributes of arrays
  - CCE requires the data attributes of every variable (including scalar variables)
  - The latest OpenACC specification comments on this:

    "Note: Any default(none) clause visible at the compute construct applies to both aggregate and scalar variables. However, any default(present) clause visible at the compute construct applies only to aggregate variables.**"**

    Solution 1: Remove all `DEFAULT(NONE)` clauses
    Solution 2: Add (first)private clauses to all compute constructs
    Solution 3: Replace every `DEFAULT(NONE)` clause by `DEFAULT(PRESENT)`

- The Cray compiler missed some of the required features that were only added in recent CCE versions

Step by step porting technique



How important are asynchronous OpenACC kernels for ICON?

- ICON most of the kernels into a separate GPU execution stream using ASYNC(1)
- This prevents unnecessary CPU/GPU synchronizations

```
!$ACC PARALLEL LOOP ASYNC(1)
DO i = 1, N
  A(i)=A(i)+A(i)*B(i)
ENDDO
!$ACC END PARALLEL

!$ACC UPDATE HOST WAIT(1)
```

	Total time
Synchronous Kernels	239 seconds
Asynchronous Kernels	227 seconds

How important is GPU based MPI communication for ICON?

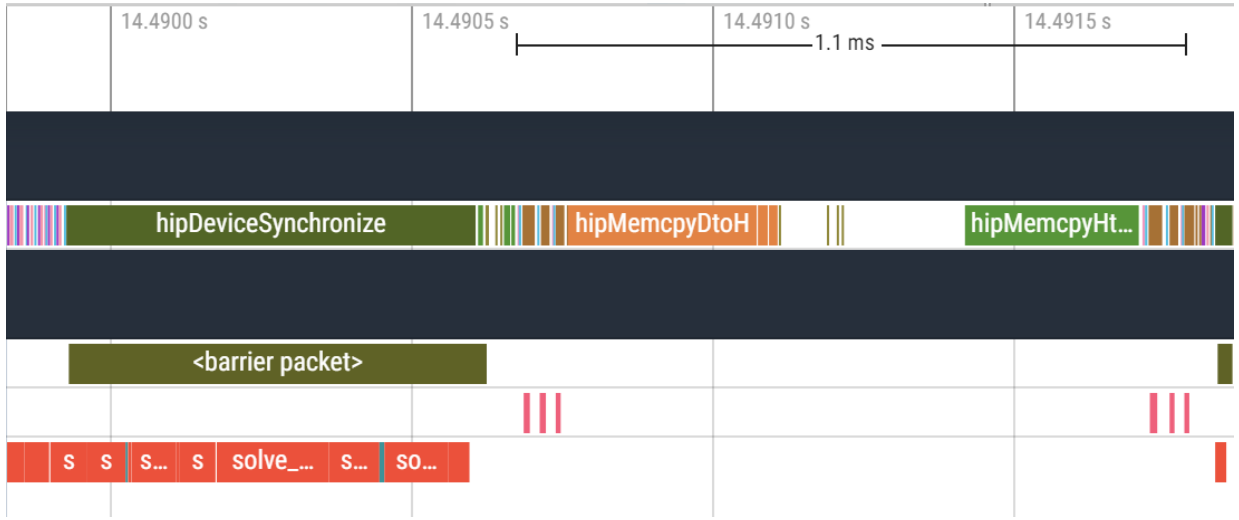
- ICON uses compiler macros to enable direct GPU to GPU communication
- How much overhead is to be expected when copying data to CPU instead?

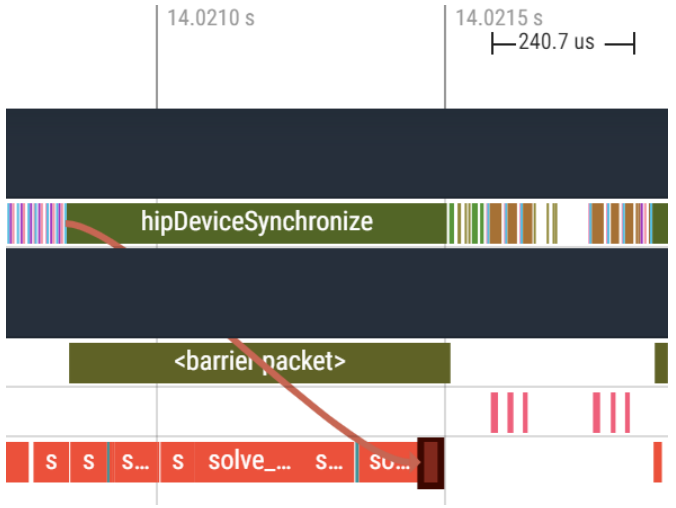
```
#ifdef __USE_G2G
!$ACC HOST_DATA USE_DEVICE(buffer)
#else
!$ACC UPDATE HOST(buffer)
#endif
    CALL mpi_send(buffer, …)
#ifdef __USE_G2G
!$ACC END HOST_DATA
#else
!$ACC UPDATE DEVICE(buffer)
#endif
```

	Total time
Communication via host	343 seconds
Device-based communication	227 seconds

AMD rocProf (rocTracer)

- rocprof is a command line tool developed by AMD for ROCProfiler and rocTracer
- hip tracing adds relatively low amount of overhead to the application (~8% for ICON)
- Visualization through chrome browser provides better understanding of the GPU interaction





AMD rocProf (rocProfiler)

- rocProfiler can profile GPU kernels with very low overhead (~4% for ICON)
- A wrapper script must be used for MPI applications
- Output is given as csv file

```
#!/bin/bash
if [ $SLURM_PROCID -eq 0 ]
then
  rocprof --stats icon
else
  icon
fi
```

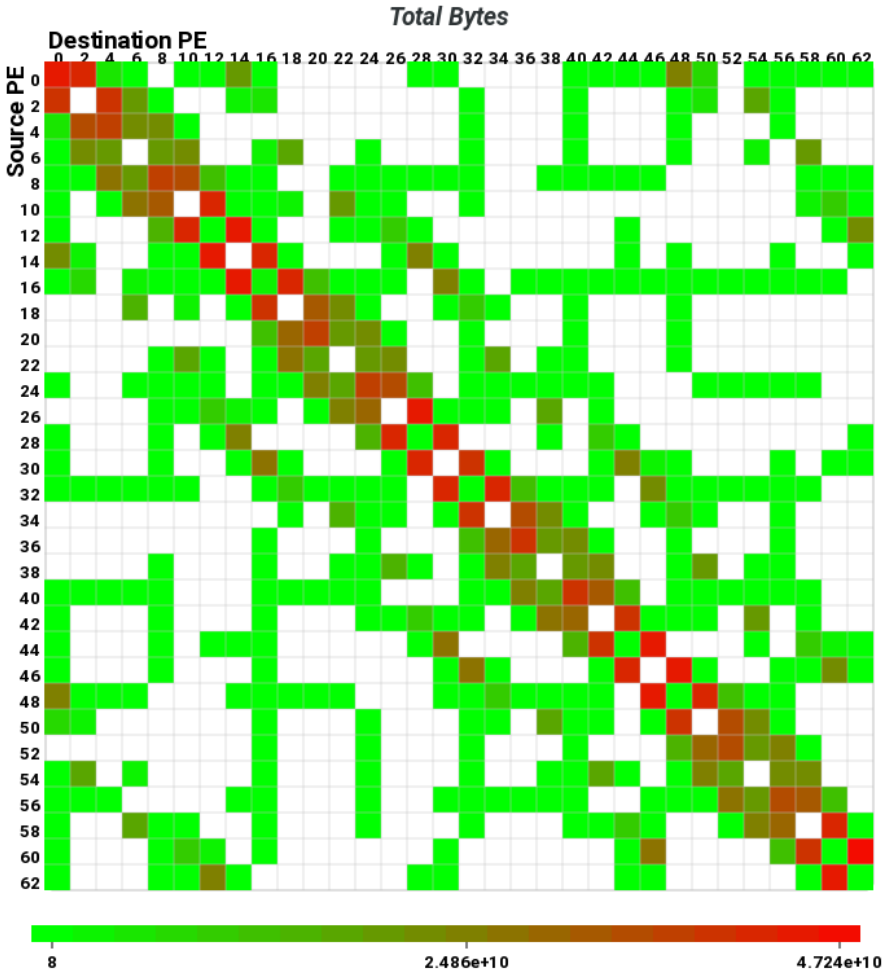
Name	Calls	TotalDurationNs	AverageNs	Percentage
solve_nh$mo_solve_nonhydro_$ck_L2400_70_cce$noloop$form.kd	43200	15659755246	362494	9.42361192
graupel$gscp_graupel_$ck_L781_4_cce$noloop$form.kd	4320	12267807125	2839770	7.38243041
rbf_vec_interpol_cell$mo_intp_rbf_$ck_L304_1_cce$noloop$form.kd	12960	6201460699	478507	3.73186924
solve_nh$mo_solve_nonhydro_$ck_L2497_75_cce$noloop$form.kd	43200	4600033150	106482	2.76817399

Cray HPE perftools-lite-gpu

- Extends the regular perftools-lite profiler by GPU analysis
- Profiling memory transfer between CPU and GPU
- Very powerful all-round tool for profiles and tracings
- High number of function and kernel calls potentially adds noticeable overhead to total runtime
- Easy usage:

```
make clean
module load perftools-lite-gpu
make
ls bin
   icon    icon+orig
```

- Visualization through Cray Apprentice2 (app2)
- ICON communication pattern shows communication with directly neighboring ranks



Helping CCE generate better code

- Weak spots in CCE's code generation were worked around in the ICON code and patched in later versions

CCE-14.0.2: 161 µs
CCE-15.0.1: 95 µs

```
!$ACC PARALLEL
!$ACC LOOP SEQ
DO j = M-1, 2, -1
  !$ACC LOOP
  DO i = N1, N2
    A(i,j)=A(i,j)+A(i,j+1)*B(i,j)
  ENDDO
ENDDO
!$ACC END PARALLEL
```

CCE-14.0.2: 84 µs
CCE-15.0.1: 83 µs

```
!$ACC PARALLEL
!$ACC LOOP
DO i = N1, N2
!$ACC LOOP SEQ
  DO j = M-1, 2, -1
    A(i,j)=A(i,j)+A(i,j+1)*B(j,j)
  ENDDO
ENDDO
!$ACC END PARALLEL
```

- The latest compilers and libraries should be used to obtain the best performance with ICON

	CCE 14.0.2	CCE 15.0.1
rocm 5.0.2	236 seconds	226 seconds
rocm 5.3.0	not available	222 seconds

Optimizing Kernels (1)

```
!$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(a,b,c,f,g) ASYNC(1)
!$ACC LOOP SEQ
DO j = 3, M
  !$ACC LOOP
  DO i = N1, N2
    g = t * p * X(i) * T(i,j) / Z(i,j)
    a = -g * B(i,j-1) * D(i,j-1)
    c = -g * B(i,j  ) * D(i,j+1)
    b = 1.0_dp + g * A(i,j) * (B(i,j-1) + B(i,j))
    f = 1.0_dp / (b + a * Q(i,j-1))
    Q(i,j) = -c * f
    Y(i,j) =  W(i,j) - g * (E(i,j-1) - E(i,j))
    Y(i,j) = (Y(i,j) - a * Y(i,j-1)) * f
  ENDDO
ENDDO
!$ACC END PARALLEL
```

Time
15.6 seconds

Optimizing Kernels (2)

```
!$ACC PARALLEL DEFAULT(PRESENT) PRIVATE(a,b,c,f,g) ASYNC(1)
!$ACC LOOP
DO i = N1, N2
  !$ACC LOOP SEQ
  DO j = 3, M
    g = t * p * X(i) * T(i,j) / Z(i,j)
    a = -g * B(i,j-1) * D(i,j-1)
    c = -g * B(i,j  ) * D(i,j+1)
    b = 1.0_dp + g * A(i,j) * (B(i,j-1) + B(i,j))
    f = 1.0_dp / (b + a * Q(i,j-1))
    Q(i,j) = -c * f
    Y(i,j) =  W(i,j) - g * (E(i,j-1) - E(i,j))
    Y(i,j) = (Y(i,j) - a * Y(i,j-1)) * f
  ENDDO
ENDDO
!$ACC END PARALLEL
```

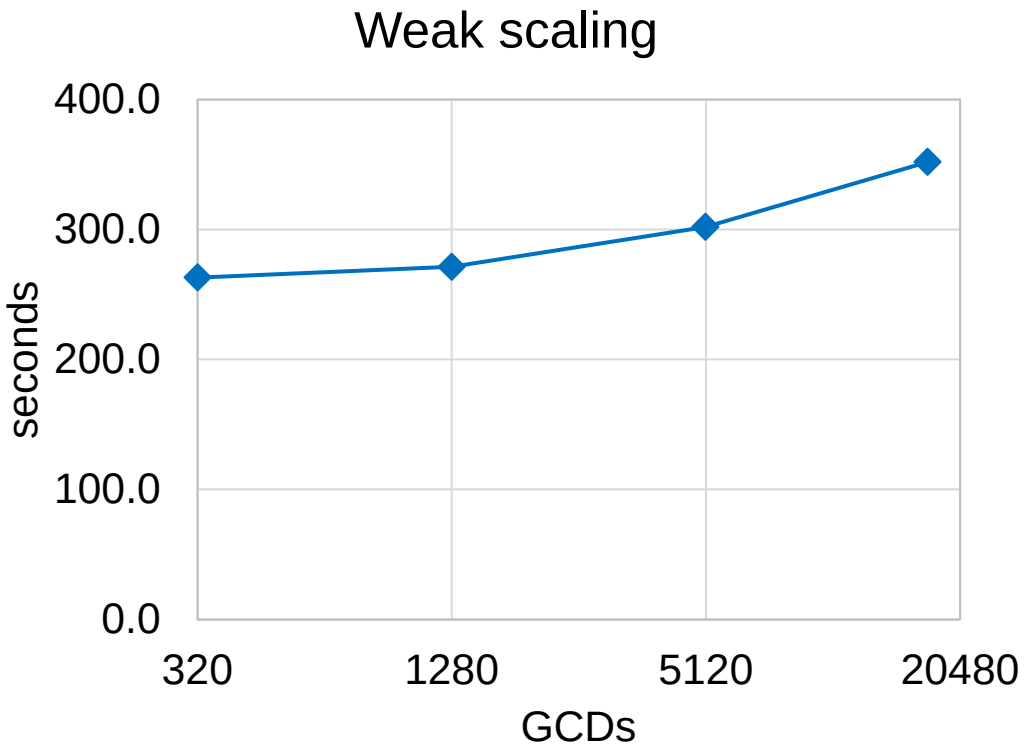
Time
13.8 seconds

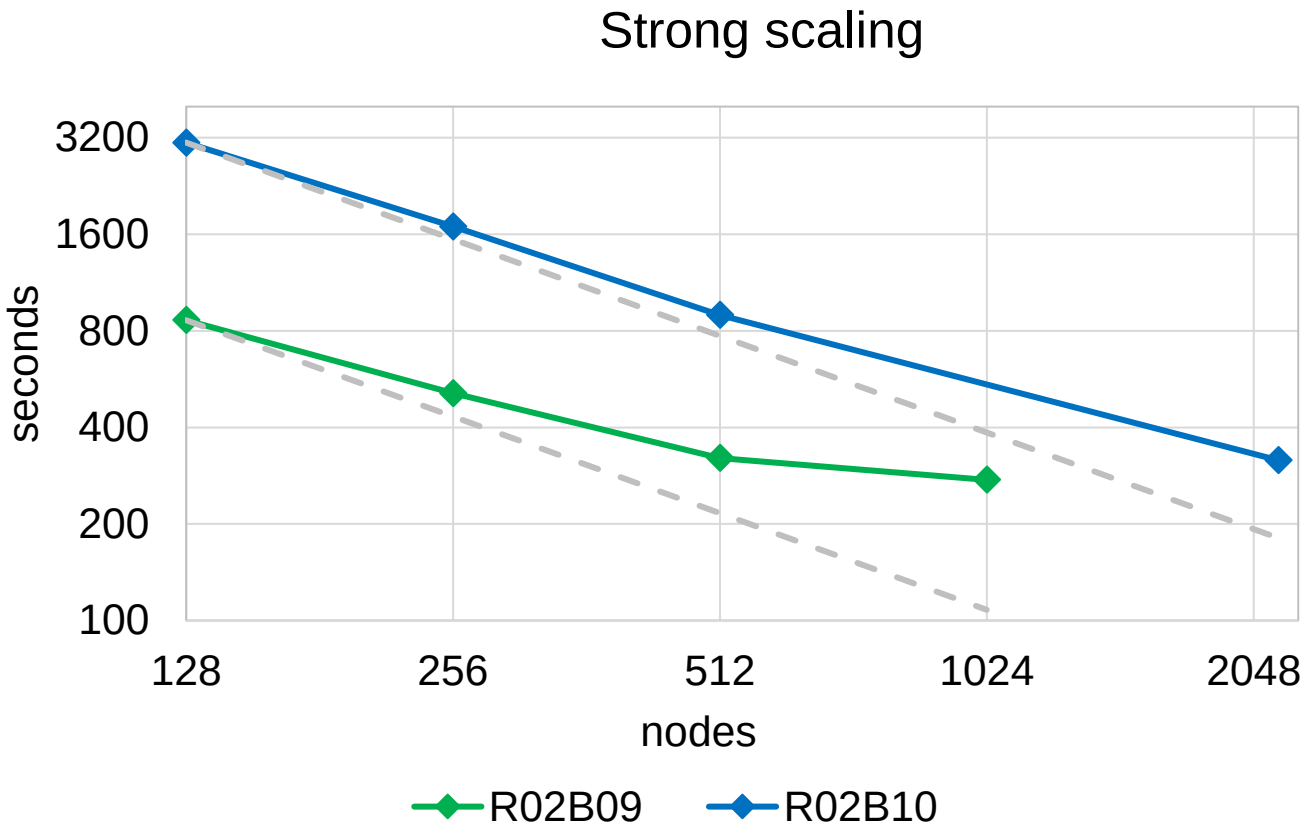
Optimizing Kernels (3)

```
!$ACC PARALLEL LOOP DEFAULT(PRESENT) PRIVATE(g) COLLAPSE(2) ASYNC(1)
DO j = 3, M
  DO i = N1, N2
    g       = t * p * X(i)* T(i,j)/Z(i,j)
    B1(i,j) = -g * B(i,j-1) * D(i,j-1)
    Q(i,j)  = g * B(i,j  ) * D(i,j+1)
    B2(i,j) = 1.0_dp + g * A(i,j) * (B(i,j-1) + B(i,j))
    Y(i,j)  = W(i,j) - g * (E(i,jk-1) - E(i,j))
  ENDDO
ENDDO
!$ACC PARALLEL LOOP PRIVATE(f) DEFAULT(PRESENT)
DO i = N1, N2
  !$ACC LOOP SEQ
  DO j = 3, M
    f      = 1.0_dp / (B2(i,j) + B1(i,j) * Q(i,j-1))
    Q(i,j) = Q(i,j) * f
    Y(i,j) = (Y(i,j) - B1(i,j) * Y(i,j-1)) * f
  ENDDO
ENDDO
```

	Time
Kernel 1	2.6 seconds
Kernel 2	8.1 seconds
Total	10.6 seconds

ICON performance evaluation on LUMI-G





Thank you

For follow up questions, feel free to contact me at peter.wauligmann@hpe.com