C2SM Center for Climate Systems Modeling

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Performance portability from a single source code

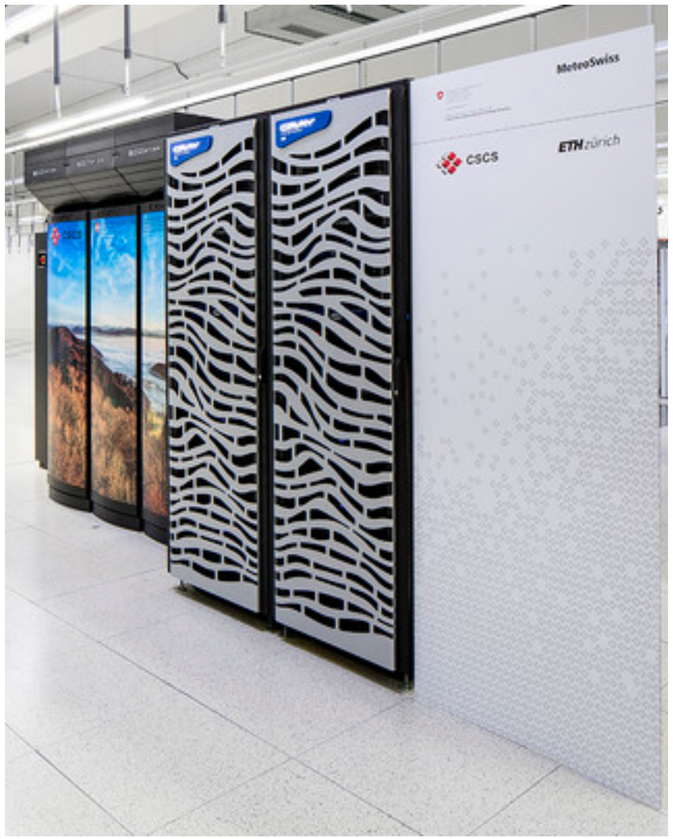
8th NCAR Multicore Workshop, National Center for Atmospheric Research, Boulder, Colorado

September 18, 2018

Jon Rood, Valentin Clement, Sylvaine Ferrachat, Oliver Fuhrer, Xavier Lapillonne, Carlos Osuna, Robert Pincus, William Sawyer

The Beginning - Performance Portability Problem

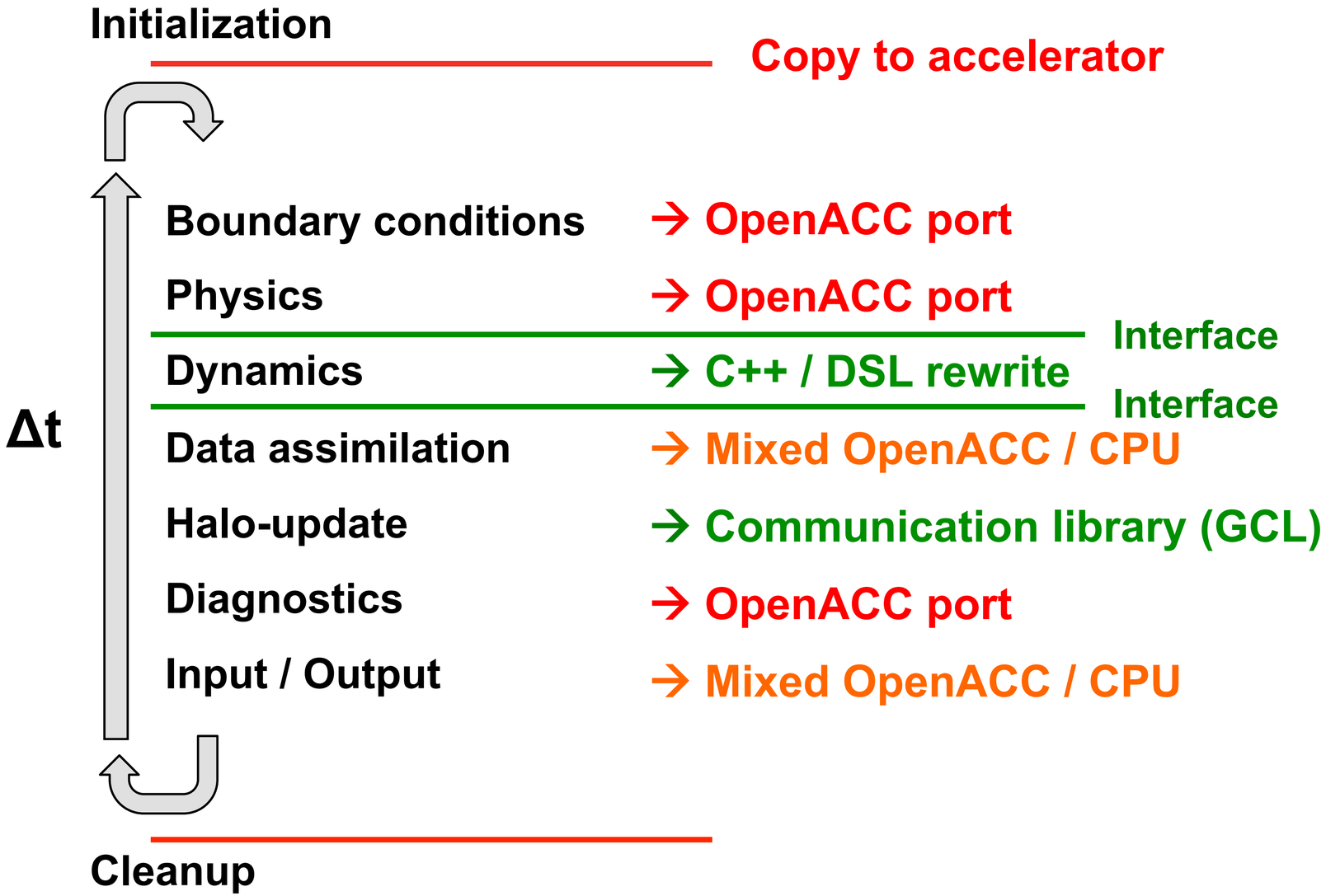
Porting COSMO to hybrid architecture



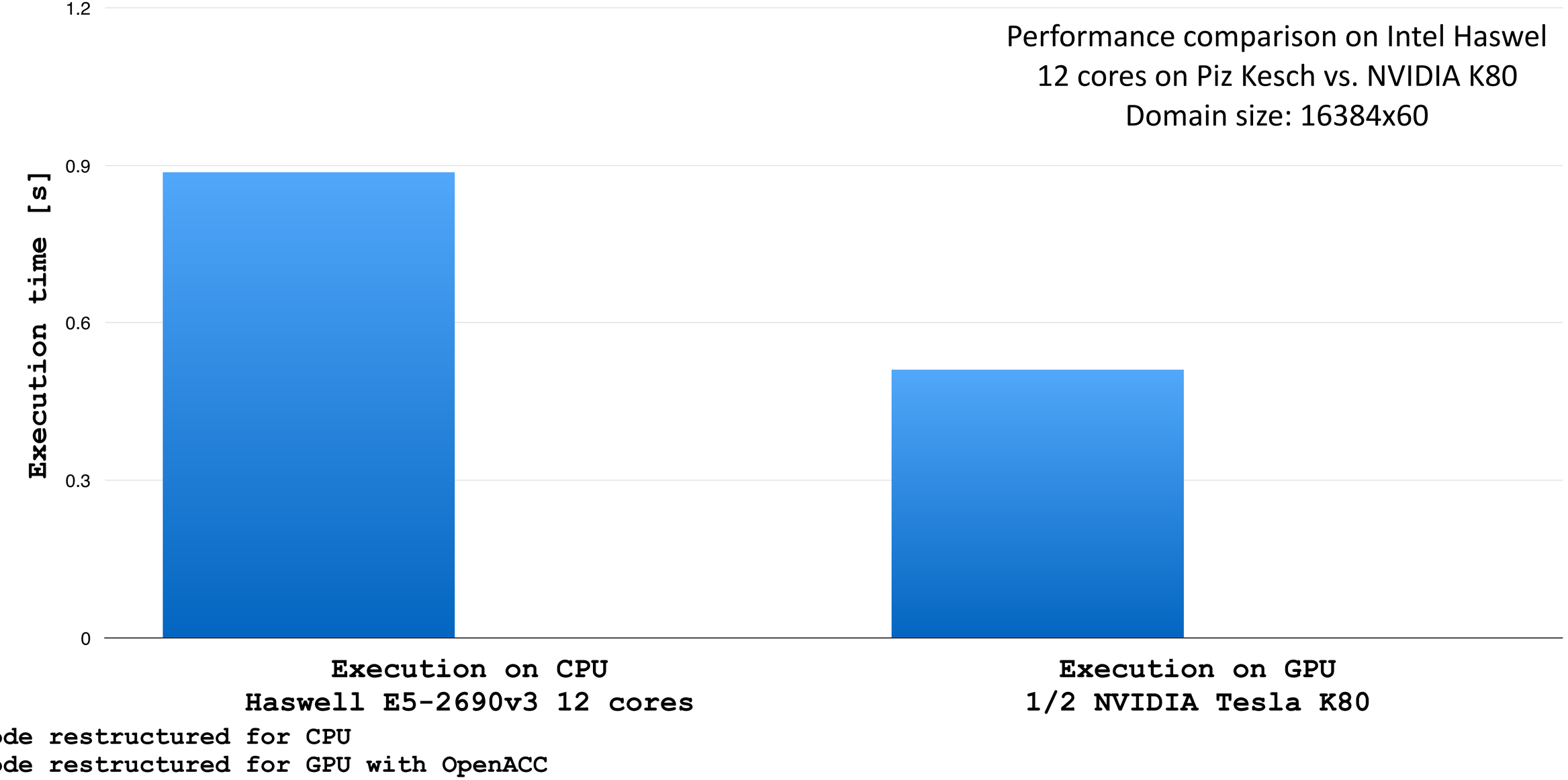
Piz Kesch - Twelve hybrid compute nodes with:

- 2 Intel Haswell E5-2690v3 2.6 GHz 12-core CPUs per node
- 8 NVIDIA Tesla K80 GPU devices per node
- 256 GB 2133 MHz DDR4 memory per node

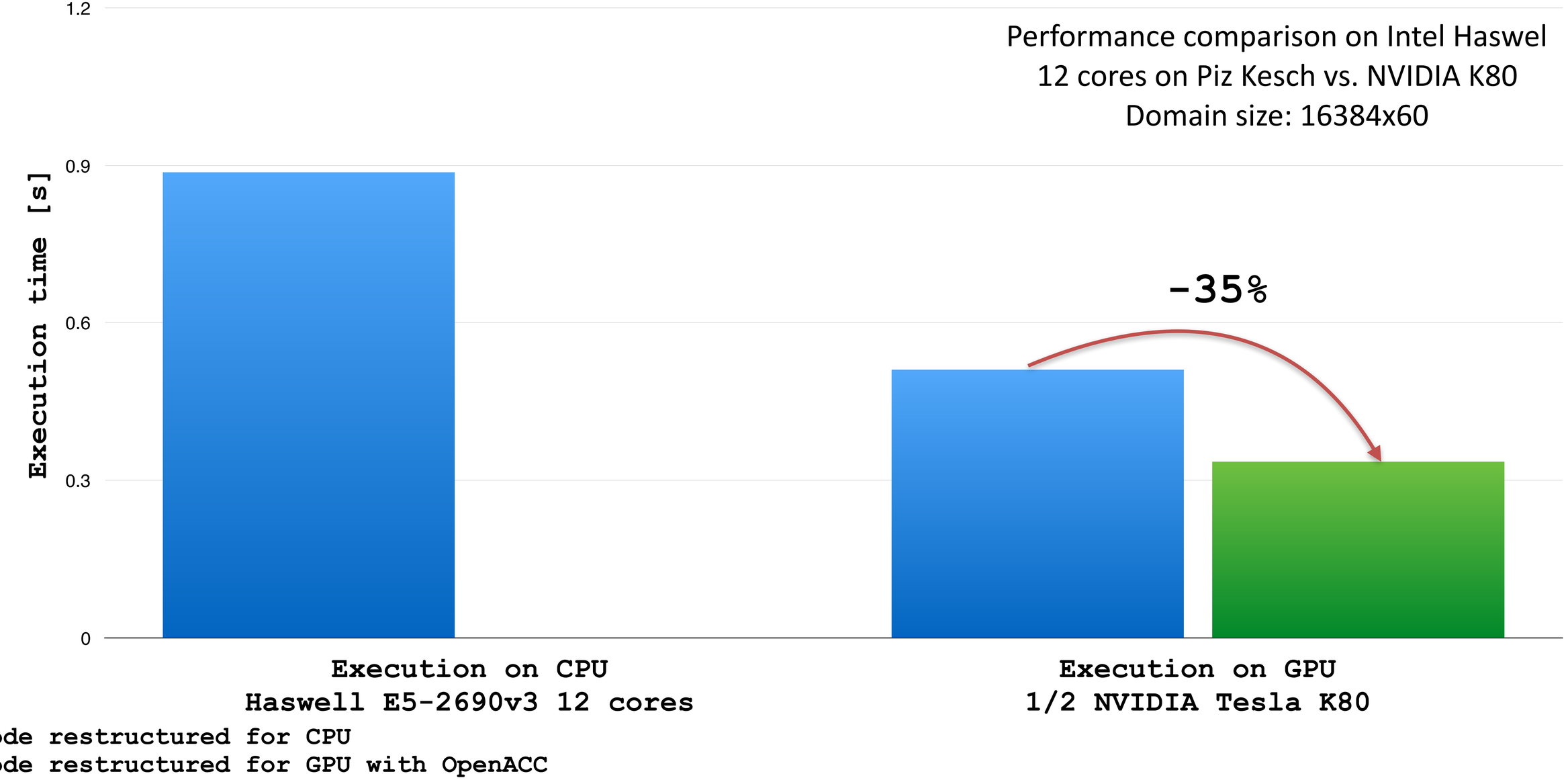
Porting COSMO to hybrid architecture



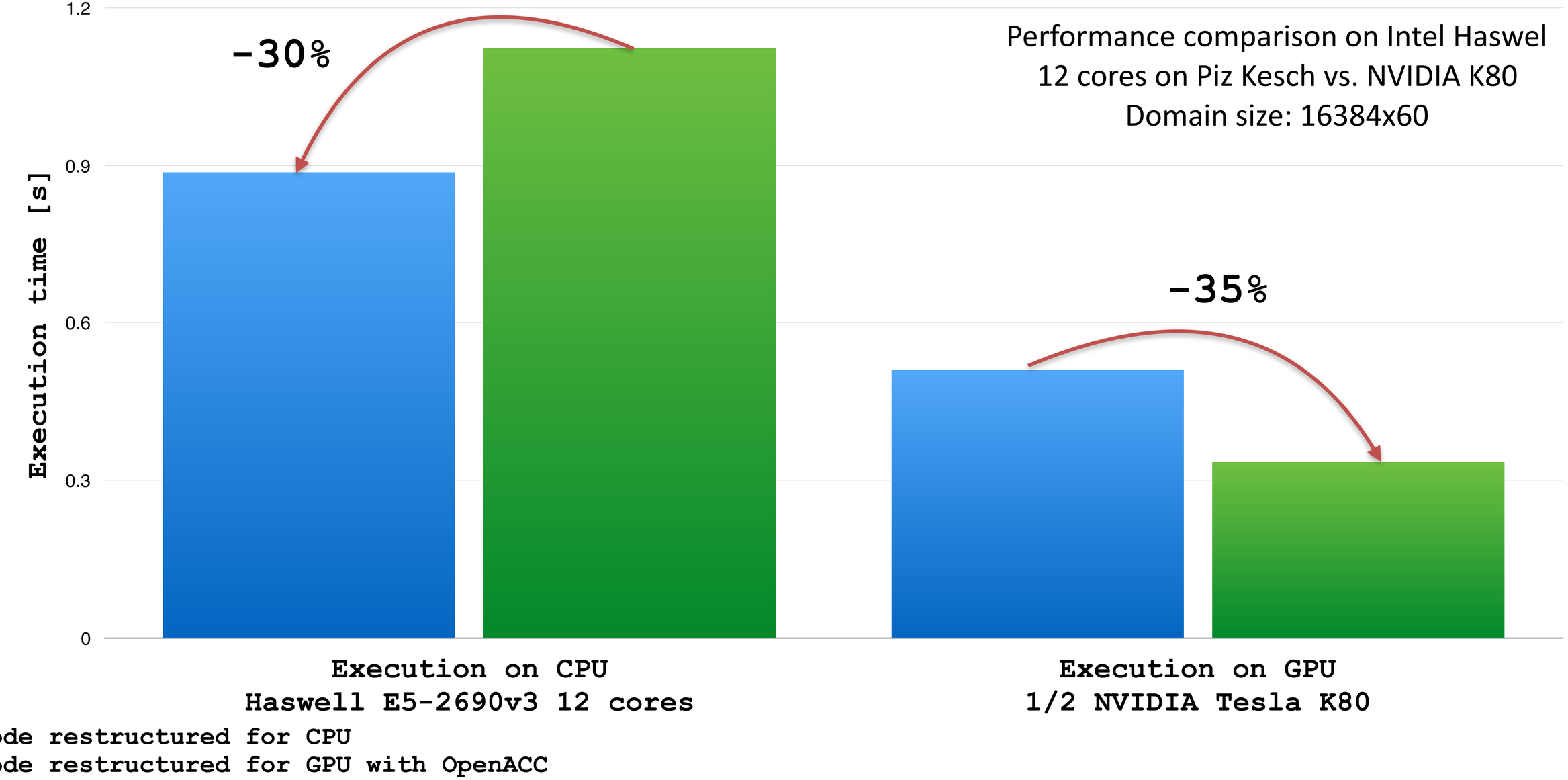
Performance portability problem - COSMO Radiation



Performance portability problem - COSMO Radiation



Performance portability problem - COSMO Radiation



Performance portability problem - COSMO Radiation

CPU structure

```
DO k=1,nz
  CALL fct()
  DO j=1,nproma
    ! 1st loop body
  END DO
  DO j=1,nproma
    ! 2nd loop body
  END DO
  DO j=1,nproma
    ! 3rd loop body
  END DO
END DO
```

GPU structure

```
!$acc parallel loop
DO j=1,nproma
  !$acc loop
  DO k=1,nz
    CALL fct()
    ! 1st loop body
    ! 2nd loop body
    ! 3rd loop body
  END DO
END DO
!$acc end parallel
```

Weather & Climate Models - One code, many users

- Several Institutes and Universities use different hardware
- Massive code base (200,000 to >1M LOC)
  - Long development cycle
  - Several architecture specific optimizations survive across versions
  - Most of these code base are CPU optimized
    - Not suited for some architectures
    - Not suited for massive parallelism
  - Software engineering: little or no modularity
    - Physical parameterization hardly extractable to the main model

Performance portability problem - Maintain multiple codes?

```
#ifndef _OPENACC
DO k=1,nz
  CALL fct()
  DO j=1,nproma
    ! 1st loop body
  END DO
  DO j=1,nproma
    ! 2nd loop body
  END DO
  DO j=1,nproma
    ! 3rd loop body
  END DO
END DO
#else
!$acc parallel loop
DO j=1,nproma
  !$acc loop
  DO k=1,nz
    CALL fct()
    ! 1st loop body
    ! 2nd loop body
    ! 3rd loop body
  END DO
END DO
!$acc end parallel
#endif
```

CPU loop structure

GPU loop structure

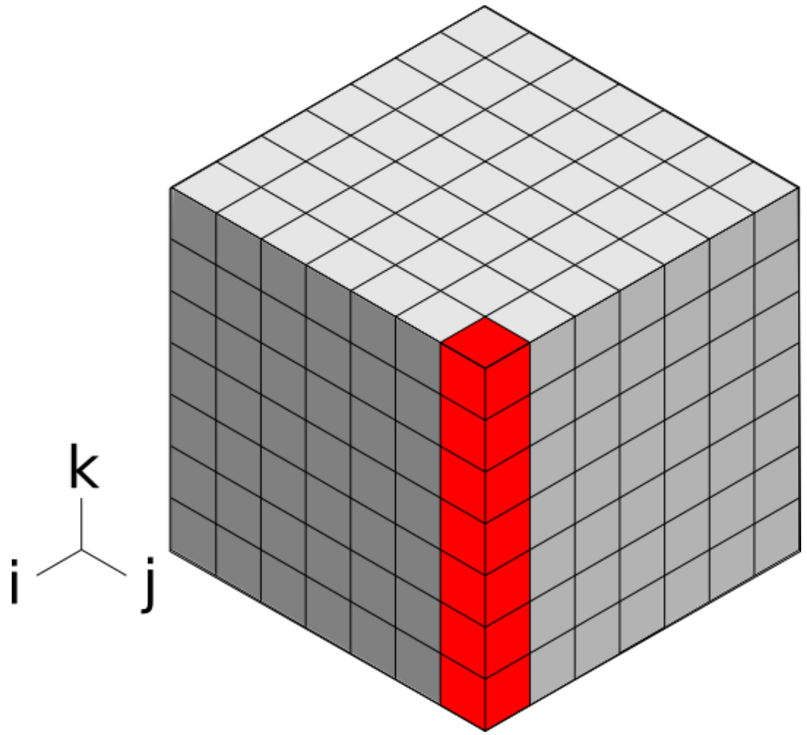
- Multiple code paths
- Difficult to maintain
- Error prone
- Domain scientists have to be experts in each target architecture

Performance portability from a single source code

- What is the best loop structure/data layout for next architecture?
- Do we want to rewrite the code each time?
- Do we have the resources to do that?
- Do we know exactly which architecture we will run on?
- Do we want to maintain a dedicated version for each architecture?

DSL - Single Column Abstraction

CLAW Single Column Abstraction (SCA)



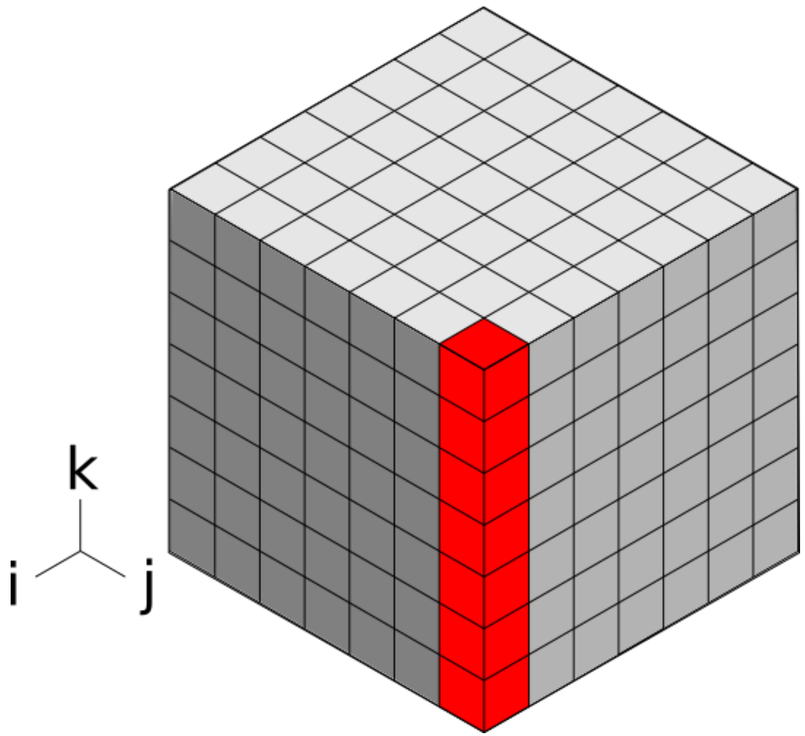
Targets physical parameterization

- Remove independent horizontal dimension
  - Remove do statements over horizontal
  - Demote arrays

Separation of concerns

- Domain scientists focus on their problem (1 column, 1 box)
- CLAW Compiler produces code for each target architecture and directive languages

RRTMGP Example - A CPU structured modular code



- F2003/F2008 Radiation Code
- From Robert Pincus and al. from AER University of Colorado
- Compute intensive part are located in “kernel” module
- Code is none-the-less CPU structured with horizontal loop as the inner most in every iteration

RRTMGP Example - original code - CPU structured

Loop over spectral bands

Loop over vertical dimension

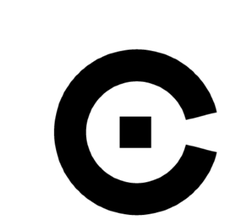
Loop over horizontal dimension

```
SUBROUTINE sw_solver(ngpt, nlay, tau, …)
 ! DECLARATION PART OMITTED
 DO igpt = 1, ngpt
   DO ilev = 1, nlay
     DO icol = 1, ncol
       tau_loc(icol,ilev) = max(tau(icol,ilev,igpt) …
       trans(icol,ilev) = exp(-tau_loc(icol,ilev))
     END DO
   END DO
   DO ilev = nlay, 1, -1
     DO icol = 1, ncol
       radn_dn(icol,ilev,igpt) = trans(icol,ilev) * radn_dn(icol,ilev+1,igpt) …
     END DO
   END DO
   DO ilev = 2, nlay + 1
     DO icol = 1, ncol
       radn_up(icol,ilev,igpt) = trans(icol,ilev-1) * radn_up(icol,ilev-1,igpt)
     END DO
   END DO
 END DO
 radn_up(:,:,:) = 2._wp * pi * quad_wt * radn_up(:,:,:)
 radn_dn(:,:,:) = 2._wp * pi * quad_wt * radn_dn(:,:,:)
END SUBROUTINE sw_solver
```

RRTMGP Example - Single Column Abstraction

Only dependency is on these iteration spaces

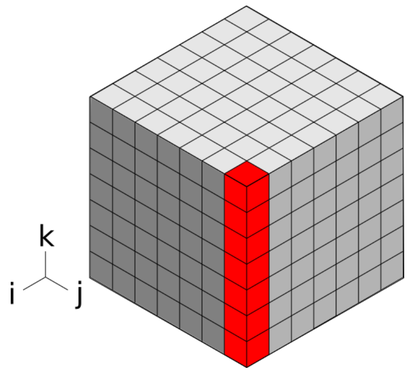
```
SUBROUTINE sw_solver(ngpt, nlay, tau, …)
  ! DECL: Fields don't have the horizontal dimension (demotion)
  DO igpt = 1, ngpt
    DO ilev = 1, nlay
      tau_loc(ilev) = max(tau(ilev,igpt) …
      trans(ilev) = exp(-tau_loc(ilev))
    END DO
    DO ilev = nlay, 1, -1
      radn_dn(ilev,igpt) = trans(ilev) * radn_dn(ilev+1,igpt) …
    END DO
    DO ilev = 2, nlay + 1
      radn_up(ilev,igpt) = trans(ilev-1) * radn_up(ilev-1,igpt)
    END DO
  END DO
  radn_up(:,:) = 2._wp * pi * quad_wt * radn_up(:,:)
  radn_dn(:,:) = 2._wp * pi * quad_wt * radn_dn(:,:)
END SUBROUTINE sw_solver
```



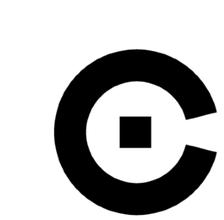
RRTMGP Example - CLAW code in subroutine

Algorithm for one column only

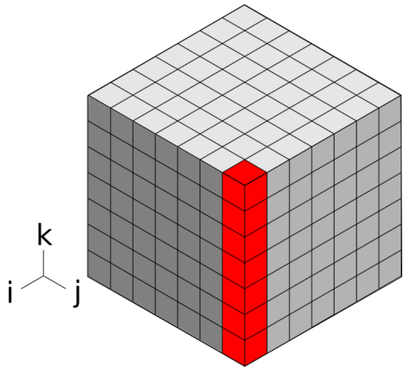
```
SUBROUTINE sw_solver(ngpt, nlay, tau, …)
 !$claw define dimension icol(1:ncol) &
 !$claw parallelize
 DO igpt = 1, ngpt
   DO ilev = 1, nlay
      tau_loc(ilev) = max(tau(ilev,igpt) …
      trans(ilev) = exp(-tau_loc(ilev))
    END DO
    DO ilev = nlay, 1, -1
      radn_dn(ilev,igpt) = trans(ilev) * radn_dn(ilev+1,igpt) …
    END DO
    DO ilev = 2, nlay + 1
      radn_up(ilev,igpt) = trans(ilev-1) * radn_up(ilev-1,igpt)
    END DO
  END DO
  radn_up(:,:) = 2._wp * pi * quad_wt * radn_up(:,:)
  radn_dn(:,:) = 2._wp * pi * quad_wt * radn_dn(:,:)
END SUBROUTINE sw_solver
```



Dependency on the vertical dimension only



RRTMGP Example - CLAW at call site




```
! Location in the model where the physical parameterization is
! plugged in

!$claw parallelize forward
DO icol = 1, ncol
   CALL sw_solver(ngpt, nlay, tau(icol,:,:), …)
END DO
```

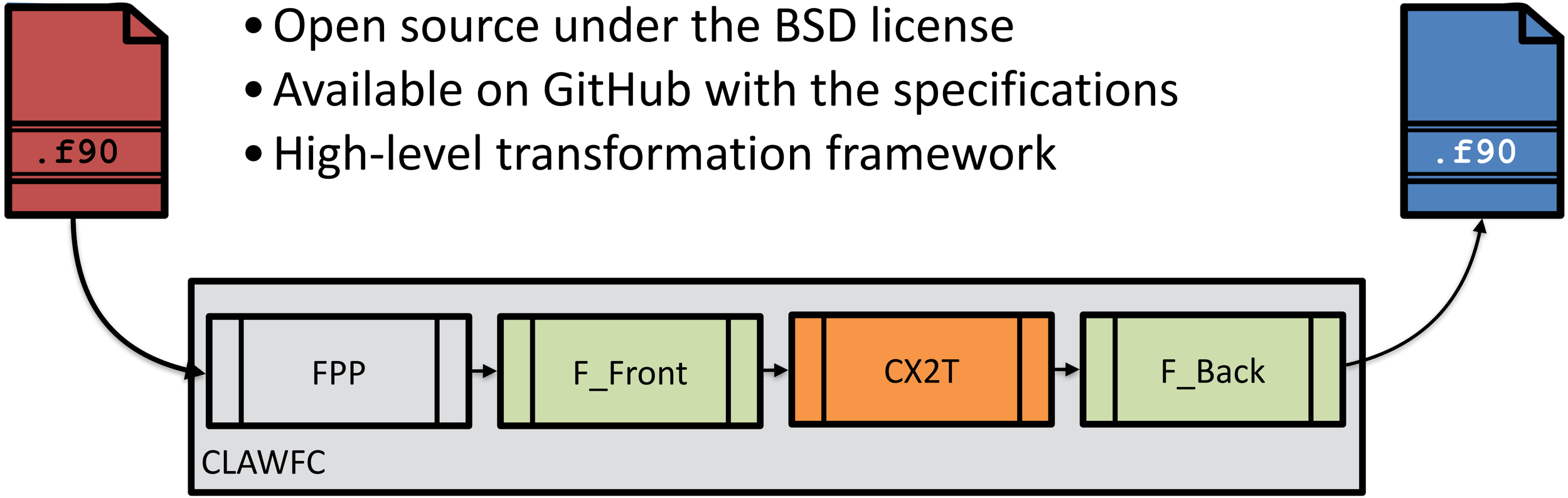
Fully working code if compiled with a standard compiler

100% standard Fortran code

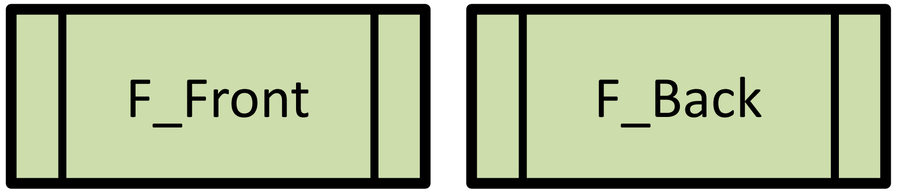
The Compiler

What is the CLAW Compiler?

- Source-to-source translator
- Based on the OMNI Compiler Project
- Fortran 2008
- Open source under the BSD license
- Available on GitHub with the specifications
- High-level transformation framework



OMNI Compiler Project



Sets of programs/libraries to build source-to-source compilers for C and Fortran via an XcodeML intermediate representation.

- XcalableMP (abstract inter-node communication), XcalableACC (XMP + OpenACC), OpenMP (implementation for C and Fortran), OpenACC (C implementation only)

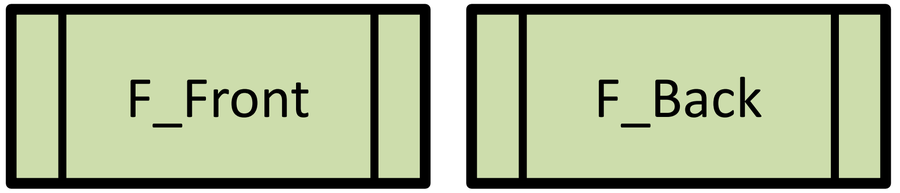
Development team

- Programming Environments Research Team from the RIKEN Center for Computational Sciences (R-CCS), Kobe, Japan
- High Performance Computing System Lab, University of Tsukuba, Tsukuba
- CLAW Project is actively collaborating in this project

http://www.omni-compiler.org
https://github.com/omni-compiler

RIKEN
Center for
Computational Science

OMNI Compiler Project



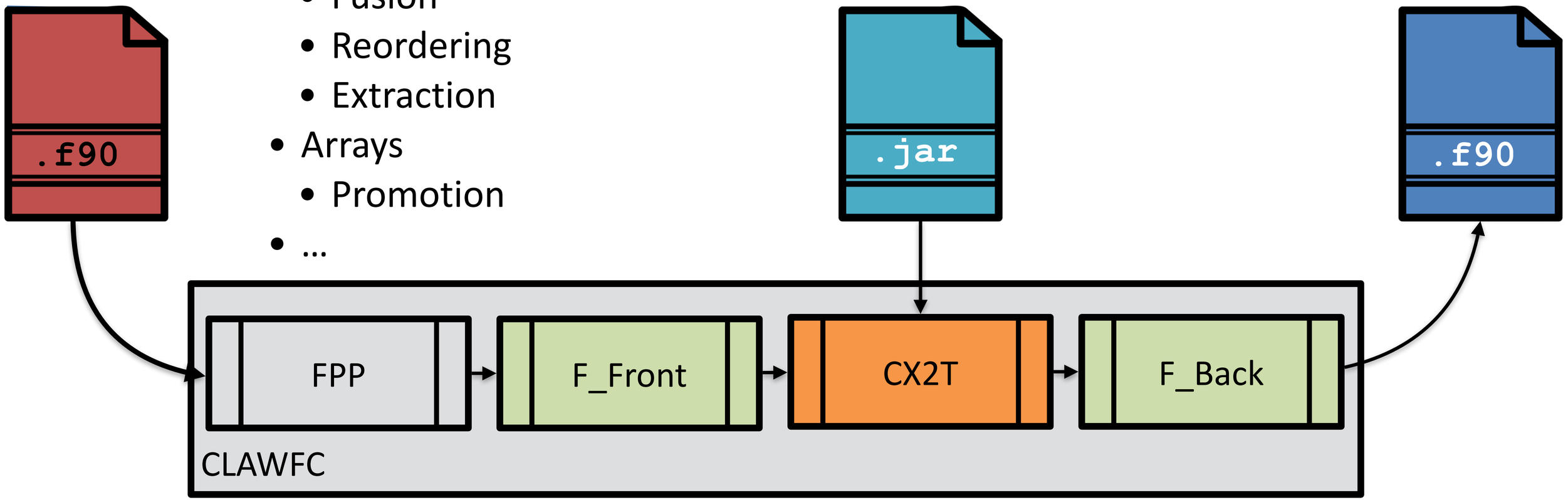
- Fortran front-end and back-end used in CLAW
- Transformations are applied on XcodeML IR
- > 100 PR contributed to OMNI Compiler from our CLAW project
- Only open-source Fortran toolchain with high-level IR able to deal with the modern Fortran code found in ICON

RIKEN
Center for
Computational Science

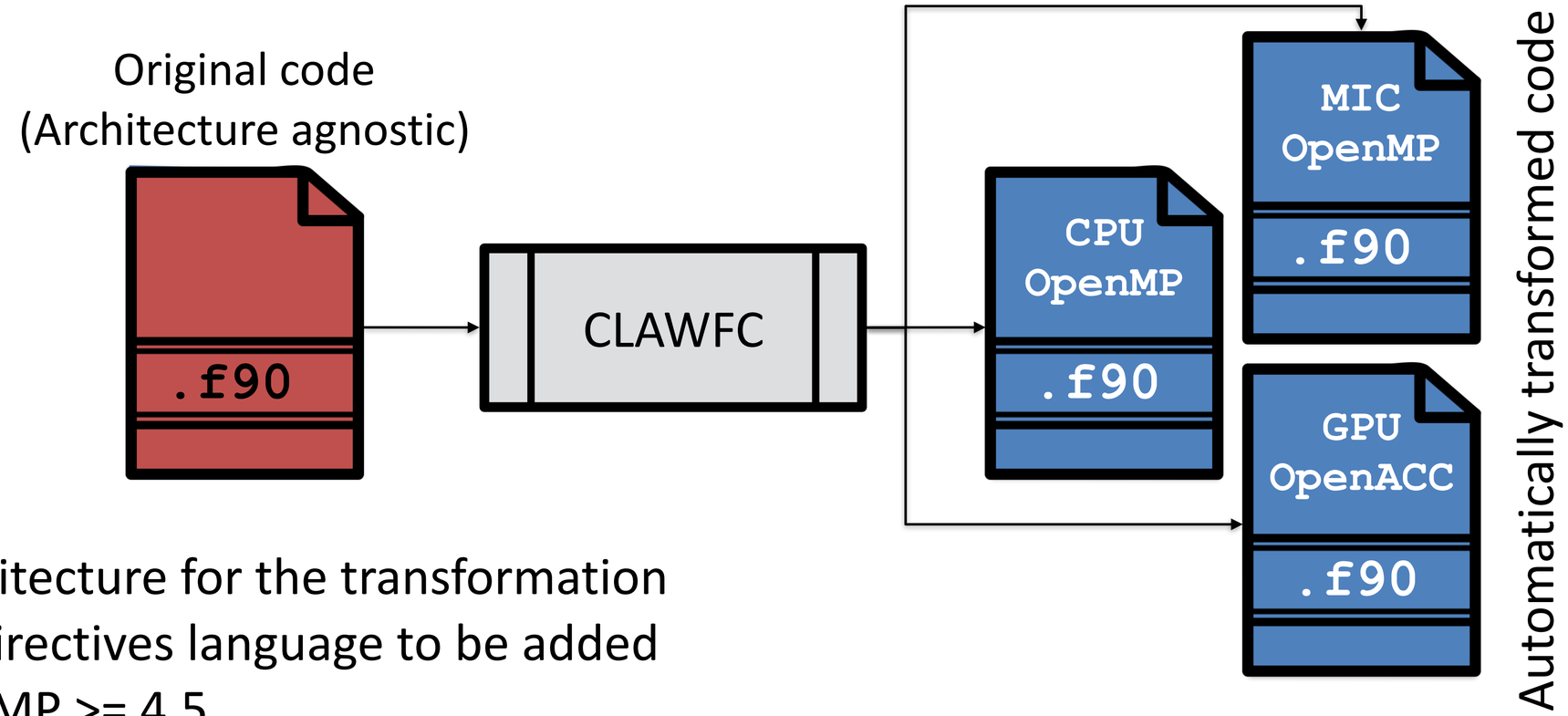
CLAW CX2T - External transformation

Easy integration of new transformation build on top of “building blocks”

- Primitive transformation
  - Loops
    - Fusion
    - Reordering
    - Extraction
  - Arrays
    - Promotion
  - …



RRTMGP Example - CLAW transformation



- A single source code
- Specify a target architecture for the transformation
- Specify a compiler directives language to be added
  - OpenACC or OpenMP >= 4.5

```
clawfc --directive=openacc --target=gpu -o mo_sw_solver.acc.f90 mo_sw_solver.f90
```

```
clawfc --directive=openmp --target=cpu -o mo_sw_solver.omp.f90 mo_sw_solver.f90
```

```
clawfc --directive=openmp --target=mic -o mo_sw_solver.mic.f90 mo_sw_solver.f90
```

CLAW SCA to target specific code - recipe

- Data dependency analysis for promotion and generation of directives
  - Potentially collapsing loops
  - Generate host to device data transfer if wanted
- Adapt data layout
  - Promotion of scalar and arrays to fit model dimensions
- Detect unsupported statements for OpenACC/OpenMP
- Insertion of do statements to iterate on new dimensions
- Insertion of directives (OpenMP/OpenACC)

CLAW Compiler has various options - example for GPU

- **Local array strategy** for Accelerator transformation
  - **Private** - issue a copy of the array for each “thread”
  - **Promote** - promote the array and keep a unique copy for all the “thread”

- **Data movement strategy** for Accelerator transformation
  - **Present** - assume that data are present on the device, no data transfer
  - **Kernel** - data movement is generated for each kernel
  - **None** - no data region generated

- **Collapse strategy** - true/false

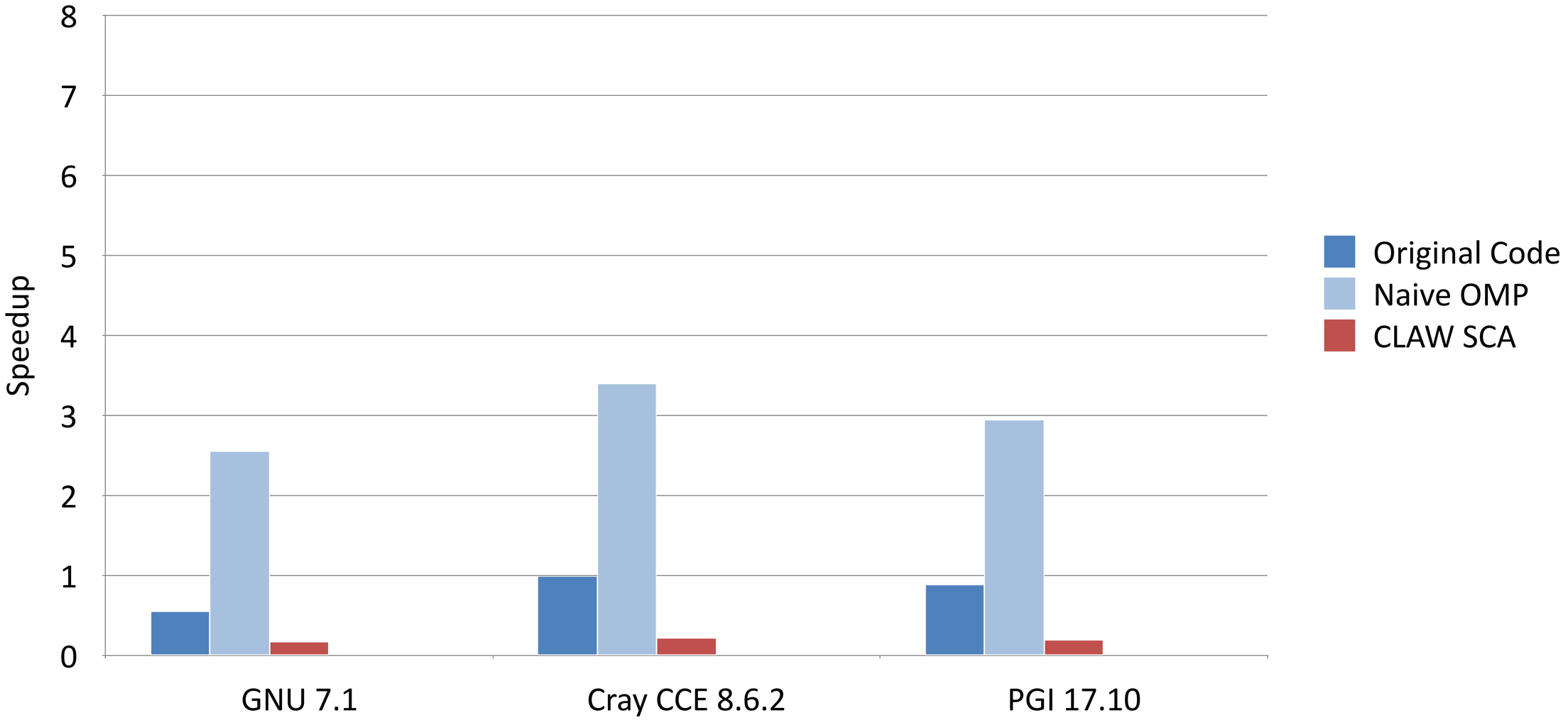
Performance Results

RRTMGP Example - CLAW target=gpu directive=openacc

```
SUBROUTINE sw_solver(ngpt, nlay, tau, …)
! DECL: Fields promoted accordingly to usage
!$acc data present(…)
!$acc parallel
!$acc loop gang vector private(…) collapse(2)
DO icol = 1 , ncol , 1
  DO igpt = 1 , ngpt , 1
    !$acc loop seq
    DO ilev = 1 , nlay , 1
      tau_loc(ilev) = max(tau(icol,ilev,igpt)
      trans(ilev) = exp(-tau_loc(ilev))
    END DO
    !$acc loop seq
    DO ilev = nlay , 1 , (-1)
      radn_dn(icol,ilev,igpt) = trans(ilev) * radn_dn(icol,ilev+1,igpt)
    END DO
    !$acc loop seq
    DO ilev = 2 , nlay + 1 , 1
      radn_up(icol,ilev,igpt) = trans(ilev-1)*radn_up(icol,ilev-1,igpt)
    END DO
  END DO
  !$acc loop seq
  DO igpt = 1 , ngpt , 1
    !$acc loop seq
    DO ilev = 1 , nlay + 1 , 1
      radn_up(icol,igpt,ilev) = 2._wp * pi * quad_wt * radn_up(icol,igpt,ilev)
      radn_dn(icol,igpt,ilev) = 2._wp * pi * quad_wt * radn_dn(icol,igpt,ilev)
    END DO
  END DO
END DO
!$acc end parallel
!$acc end data
END SUBROUTINE sw_solver
```

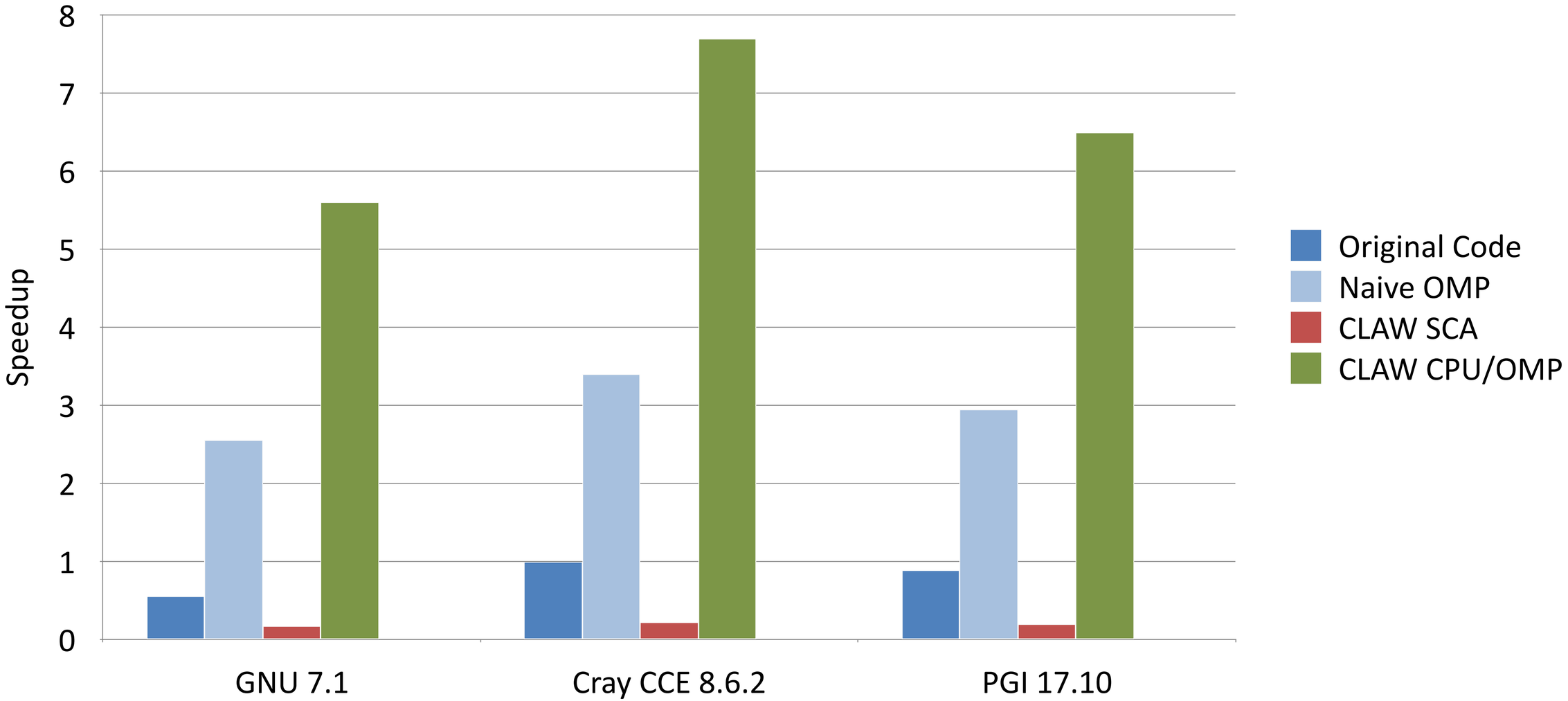
RRTMGP Example - Speedup on CPU

Performance comparison on Intel Xeon E5-2690 v3 - 1 core vs. 12 cores on Piz Daint
Domain size: 16384x42 + 14 spectral bands



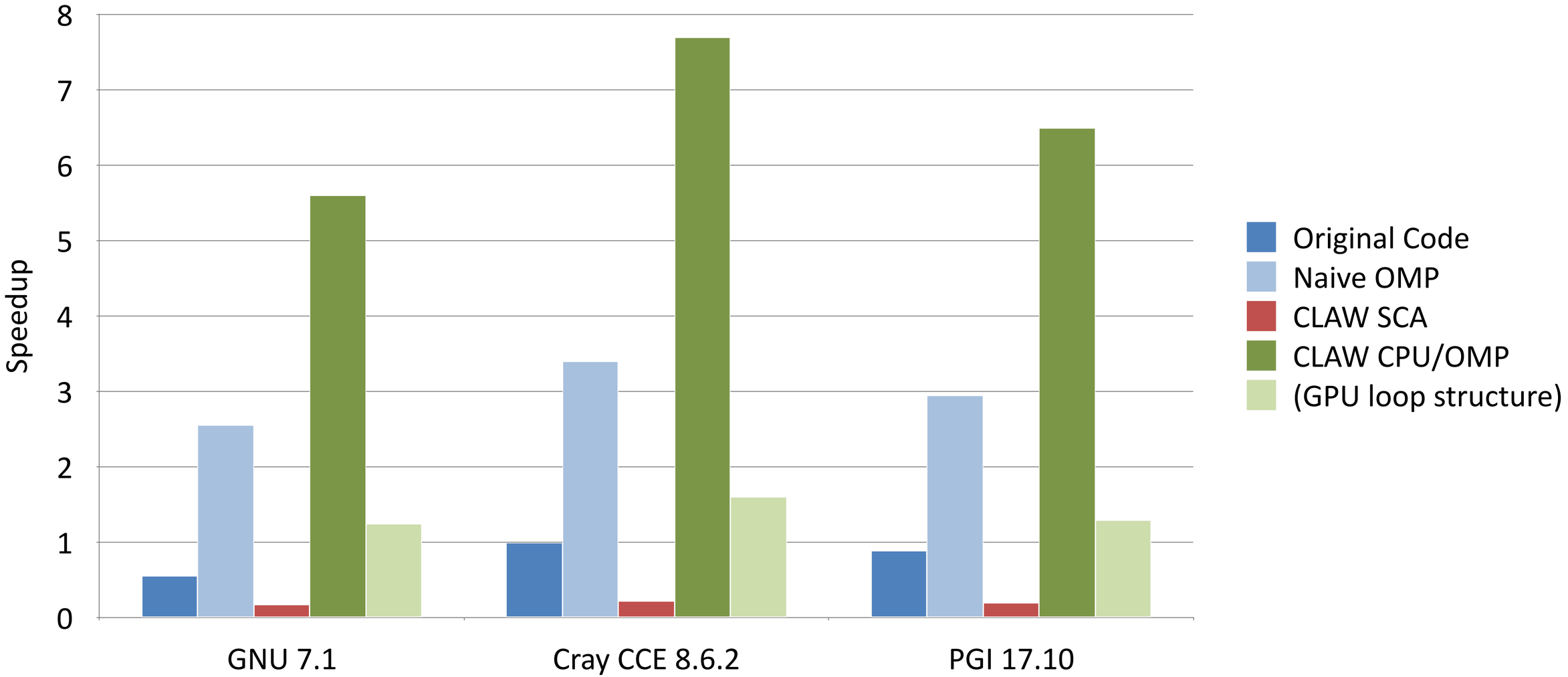
RRTMGP Example - Speedup on CPU

Performance comparison on Intel Xeon E5-2690 v3 - 1 core vs. 12 cores on Piz Daint
Domain size: 16384x42 + 14 spectral bands

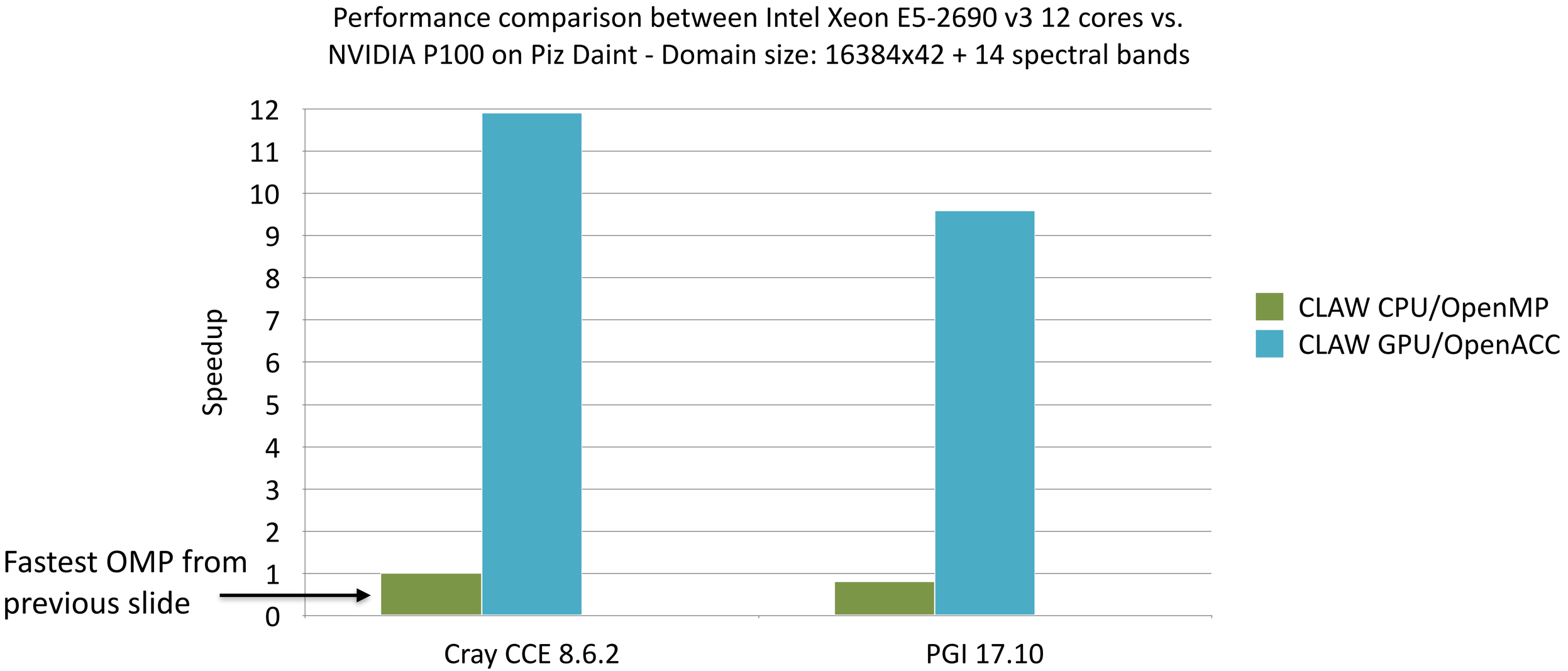


RRTMGP Example - Speedup on CPU

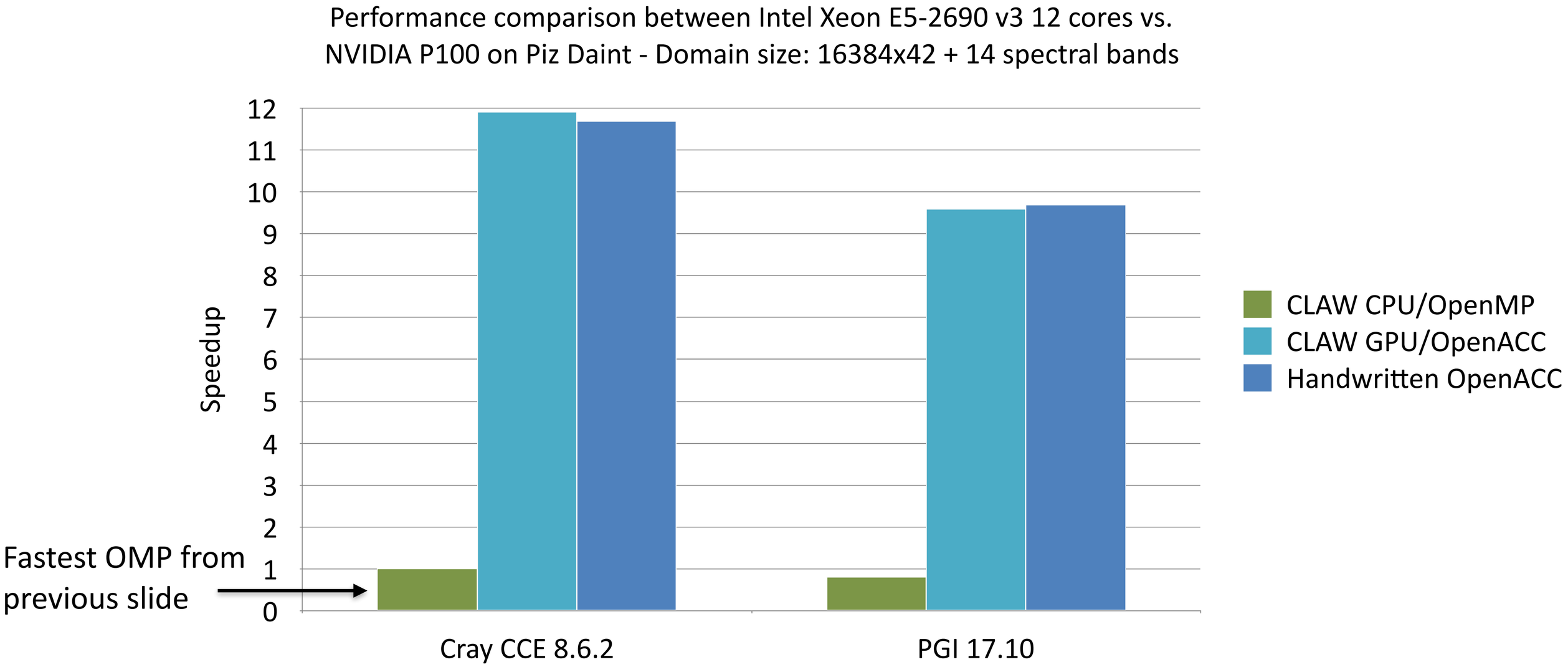
Performance comparison on Intel Xeon E5-2690 v3 - 1 core vs. 12 cores on Piz Daint
Domain size: 16384x42 + 14 spectral bands



RRTMGP Example - Speedup CPU vs. GPU



RRTMGP Example - Speedup CPU vs. GPU



Code metrics

	sw_solver
Demoted Arrays	35
Removed do statments	15
CLAW directive	3

81% of the code is kept from original

Applied in micro-physics from ICON
CLAW GPU/OpenACC and CLAW CPU/OpenMP versions reach similar performance from a hand-written version

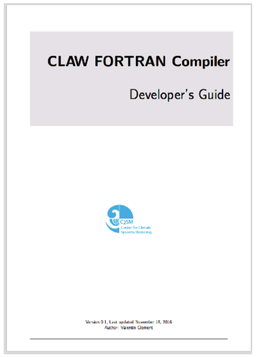
PASC ENIAC Project (2017-2020)

- Enabling ICON model on heterogenous architecture
  - Port to OpenACC
  - GridTools for stencil computation (DyCore)
  - Looking at performance portability in Fortran code
    - Enhance CLAW Compiler capabilities
    - Apply SCA on some physical parameterization
    - Enhance transformation for x86, Xeon Phi and GPUs

CLAW Compiler & Directives - Resources

https://claw-project.github.io

https://github.com/omni-compiler



CLAW Compiler developer’s guide

Summary

- Single source code with high-level of abstraction
- Domain scientists able to focus on their problem
- Little change to current code
- Standard Fortran
- Open source project
- CLAW is easily extensible to new architectures or new transformations

C2SM Center for Climate Systems Modeling

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

contact: valentin.clement@env.ethz.ch

Valentin Clement, Sylvaine Ferrachat, Oliver Fuhrer, Xavier Lapillonne, Carlos Osuna, Robert Pincus, Jon Rood, William Sawyer

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