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

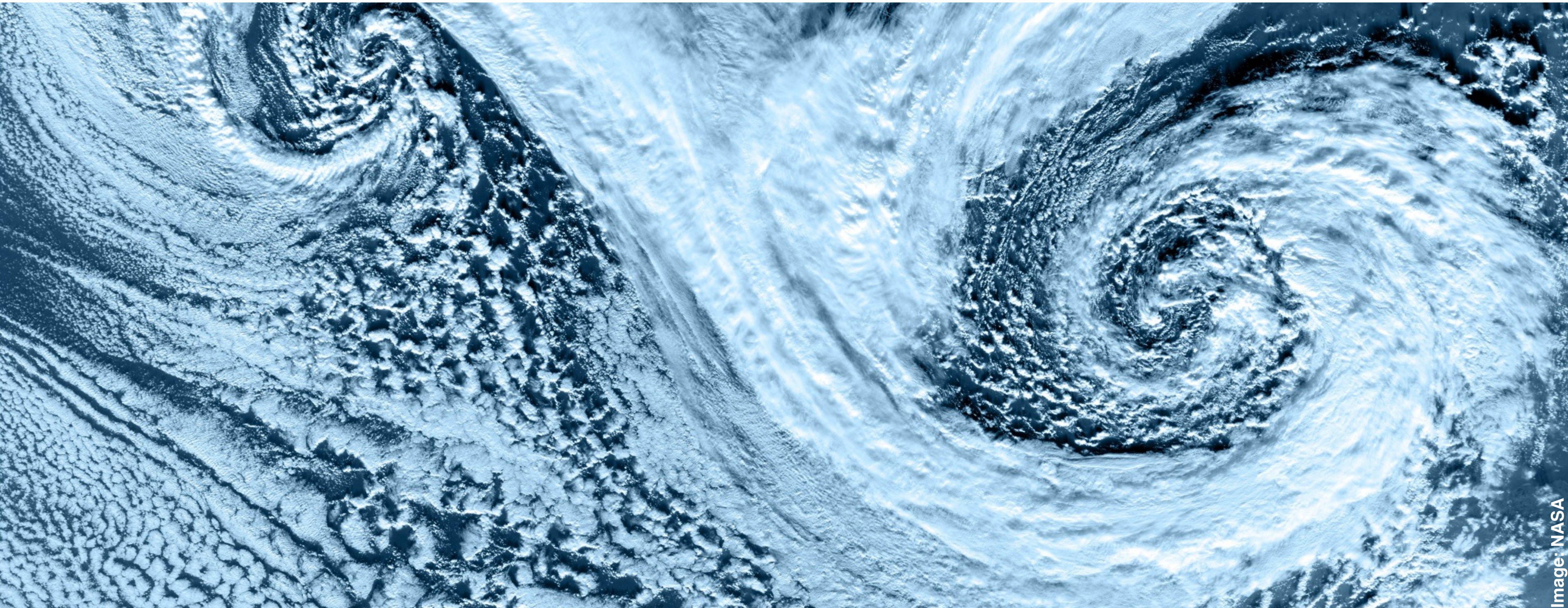
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CLAW One code to rule them all

4th XcalableMP Workshop, Tokyo, Japan
November 7, 2016

Valentin Clement
valentin.clement@env.ethz.ch



Summary

- Weather prediction model to new architecture
- Performance portability problem
- CLAW low-level transformations
- CLAW high-level abstraction
- CLAW with OMNI as source-to-source translator
- Other initiative at MeteoSwiss

C2SM
Center for Climate
Systems Modeling

COSMO at MeteoSwiss

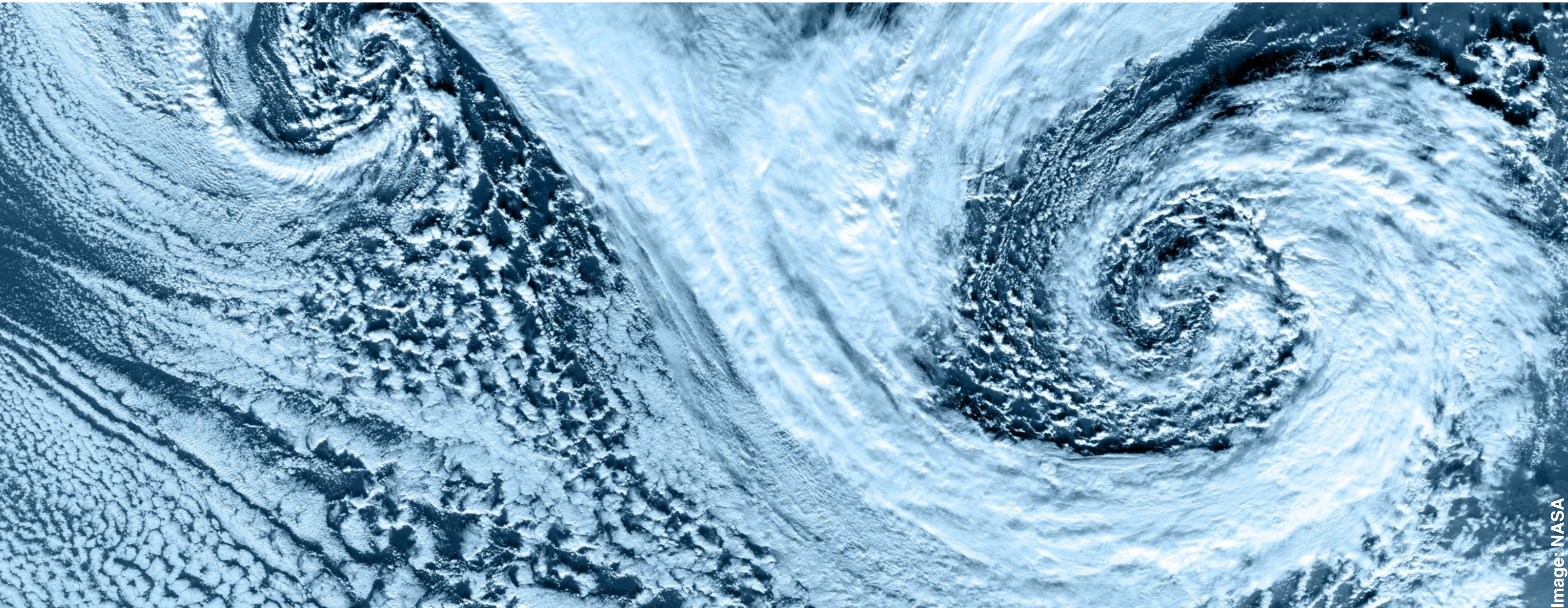


Image: NASA

COSMO

- Non-hydrostatic limited-area atmospheric prediction model
- Develop by community of the COSMO consortium
- **FORTRAN** is the main language
- Previously only target CPU architecture
- COSMO Consortium
  - 7 national weather services
  - Universities

COSMO previous operation setup at MeteoSwiss

ECMWF-Model

16 km gridspacing

2 x per day 10 day forecast

COSMO-7

6.6 km gridspacing

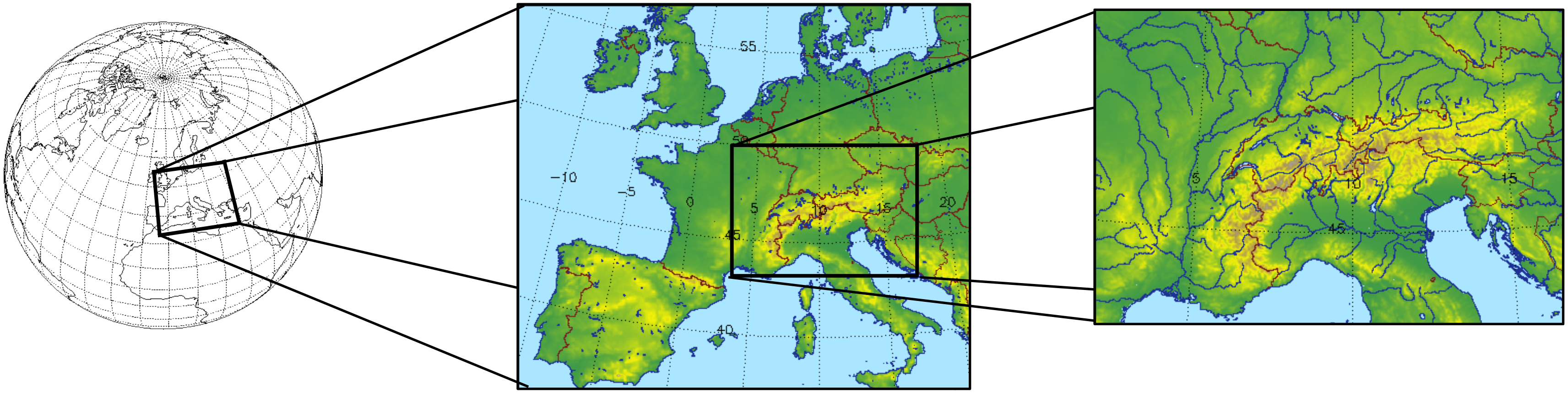
3 x per day 72 h forecast

COSMO-2

2.2 km gridspacing

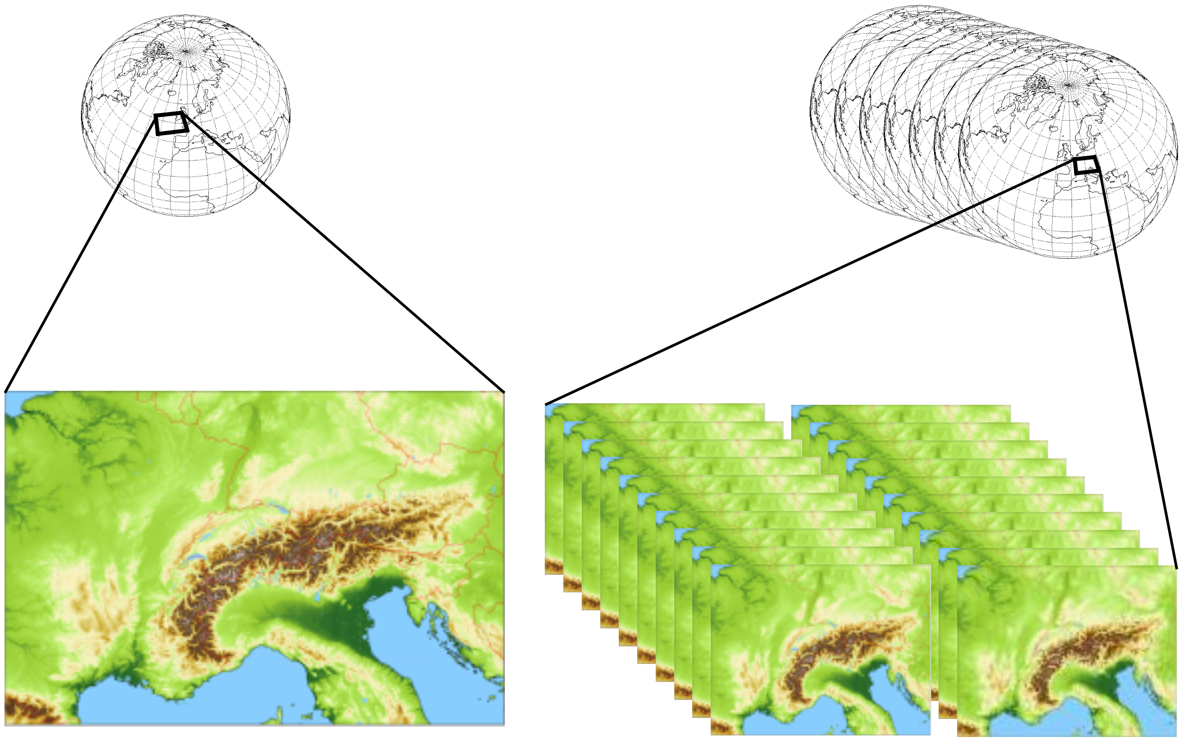
7 x per day 33 h forecast

1 x per day 45 h forecast



COSMO New operation setup at MeteoSwiss

COSMO-1
1.1 km gridspacing
8 x per day
1 to 2 d forecast

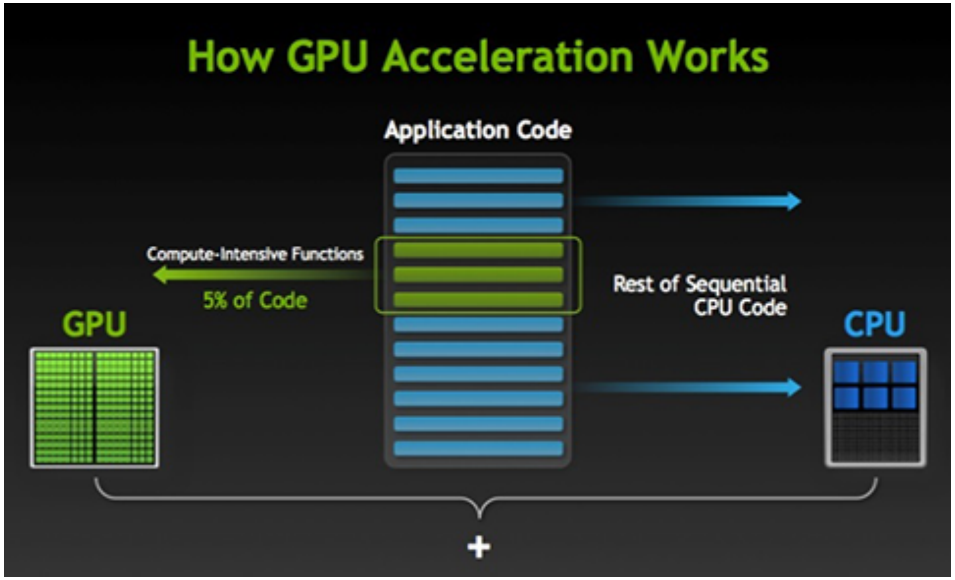


COSMO-E
2.2 km gridspacing
2 x per day
5 d forecast
21 members

Computational cost = 40x

(compare to previous operational system)

Hybrid supercomputer (CPU/GPU)



Applications

Libraries

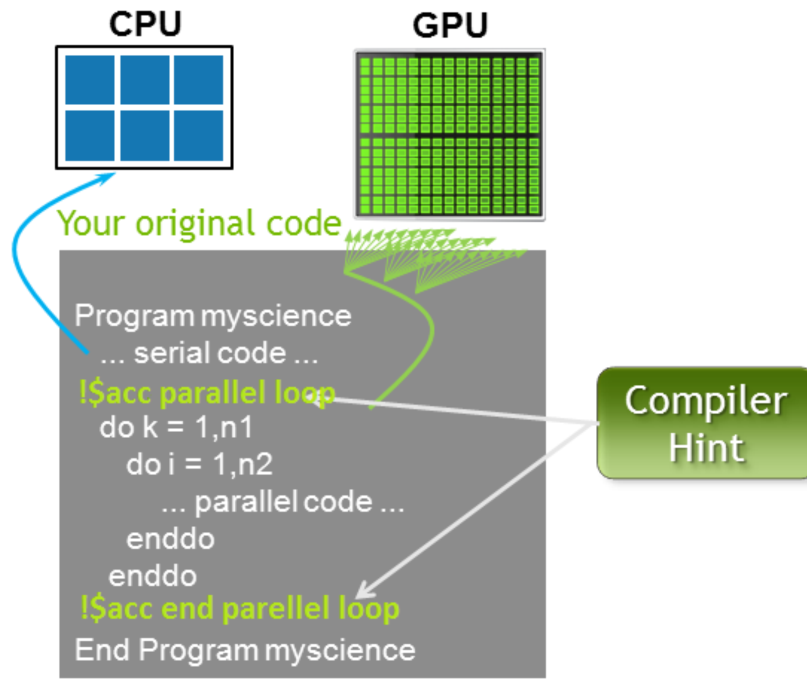
Compiler Directives

Programming Languages and DSLs

Ex: Magma, CULA, cuBLAS
...

Ex : OpenACC, OpenMP 4.0

Cuda, Cuda Fortran, OpenCL, STELLA



COSMO to the Hybrid supercomputers

Initialization

Copy to accelerator

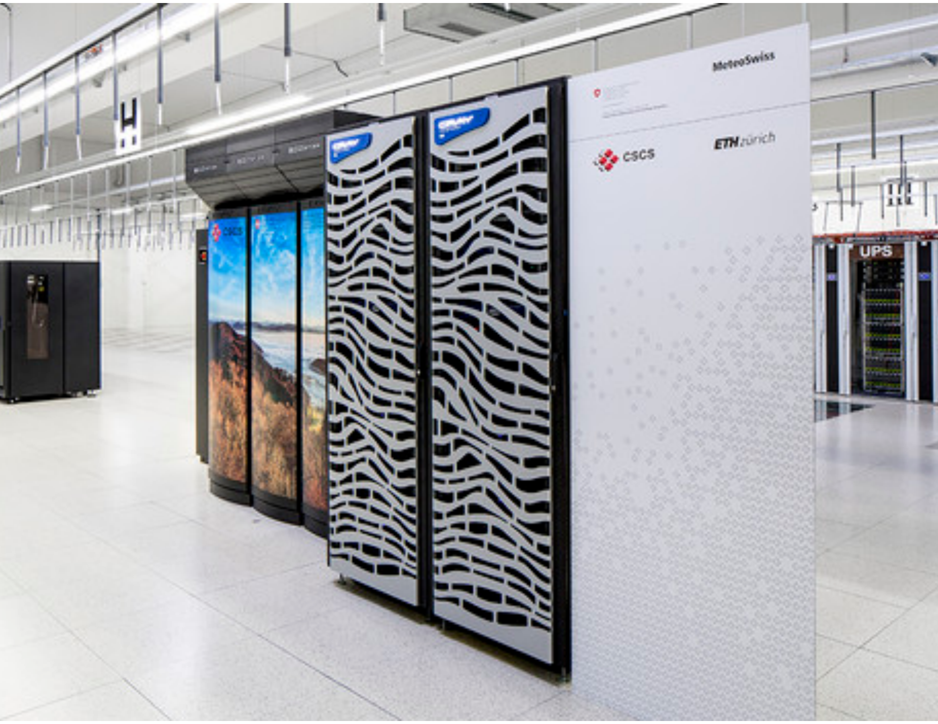
Δt

Boundary conditions	→ OpenACC port
Physics	→ OpenACC port
Interface	
Dynamics	→ C++ / DSL rewrite
Interface	
Data assimilation	→ Mixed OpenACC / CPU
Halo-update	→ Communication library (GCL)
Diagnostics	→ OpenACC port
Input / Output	→ Mixed OpenACC / CPU

Cleanup

C2SM Center for Climate Systems Modeling

Kesch & Escha - MeteoSwiss



Production + R&D

Each rack is composed of 12 Compute nodes:

- 2x Intel Haswell E5-2690v3
  2.6 GHz 12-core CPUs (total of 24 CPUs)
- 256 GB 2133 Mhz DDR4 (total of 3TB)
- 8x Dual NVIDIA TESLA K80 GPU (total of 96 cards - 192 GPUs)

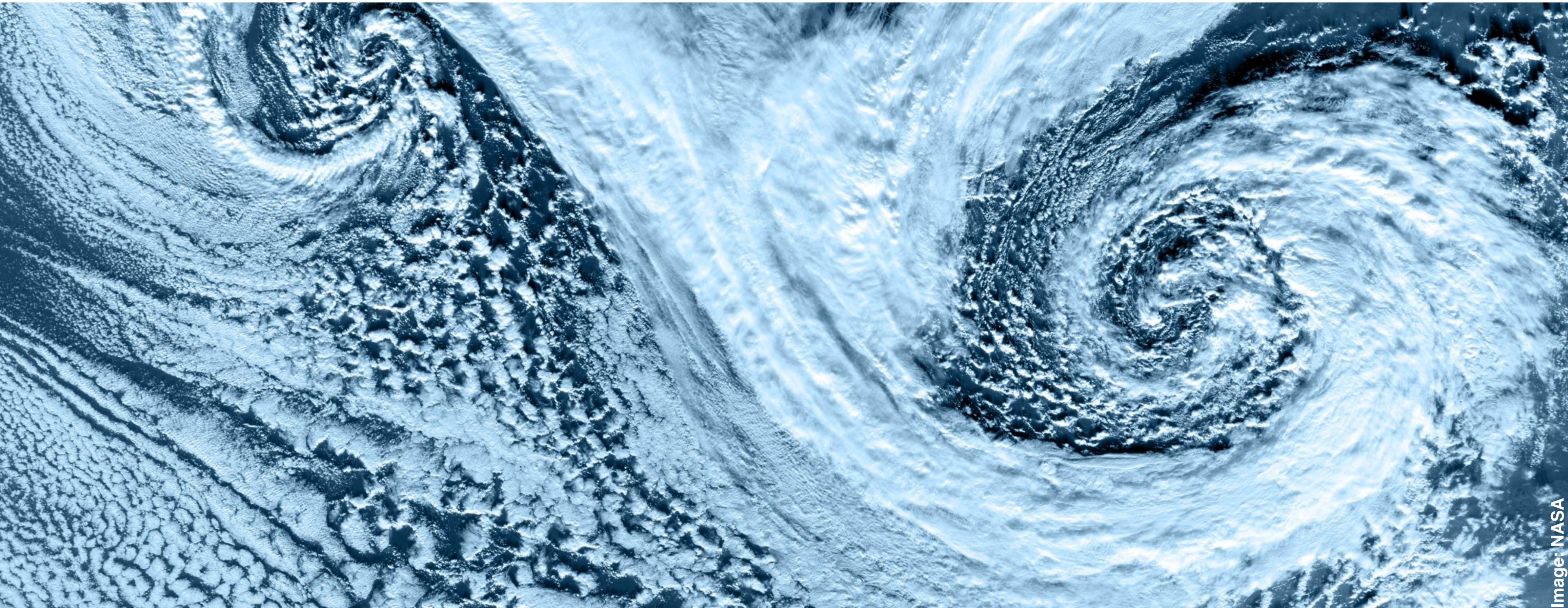
Piz Daint - CSCS users supercomputer

Cray XC30 - 5272 computes nodes

- 1x Intel Xeon E5-2670
- 1x NVIDIA Tesla K20X
- 32 GB 2133 Mhz DDR4 (total of 169TB)

Currently upgraded to **Intel Haswell CPU** and **PASCAL GPU**

The performance portability problem



Performance portability

Portable between what?

- Between different CPUs?
- Between different compilers for the same architecture?
- Between different architectures CPU vs CPU/GPU vs MIC?

"Most people are resigned to having different sources for different platforms, with simple #ifdef or other mechanisms"
DOE workshop output on performance portability

Can we do it better for our FORTRAN code than #ifdef?

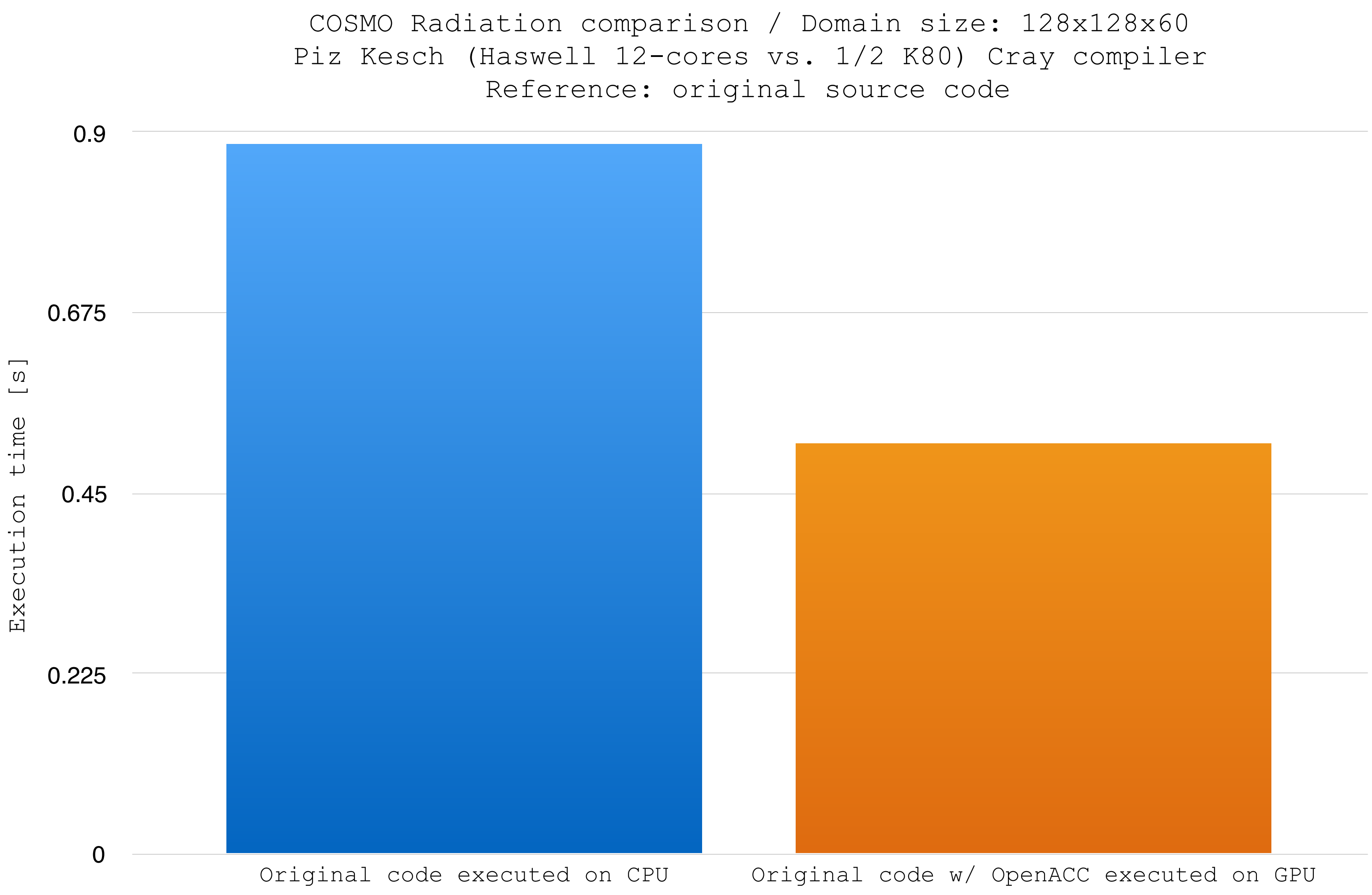
COSMO radiation example

Iteration over 2 dimensions (j = horizontal, k = vertical)

```
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
```

Typical CPU optimal code structure

Performance portability: COSMO Radiation



Performance portability

In some cases CPU and GPU have different optimization requirements

CPU:

- Auto-vectorization: small loops
- Pre-computation

GPU:

- Benefit from large kernels : reduce kernel launch overhead, bett computation/memory access overlap
- Loop re-ordering and scalar replacement
- On the fly computation

COSMO radiation example

Iteration over 2 dimensions (j = horizontal, k = vertical)

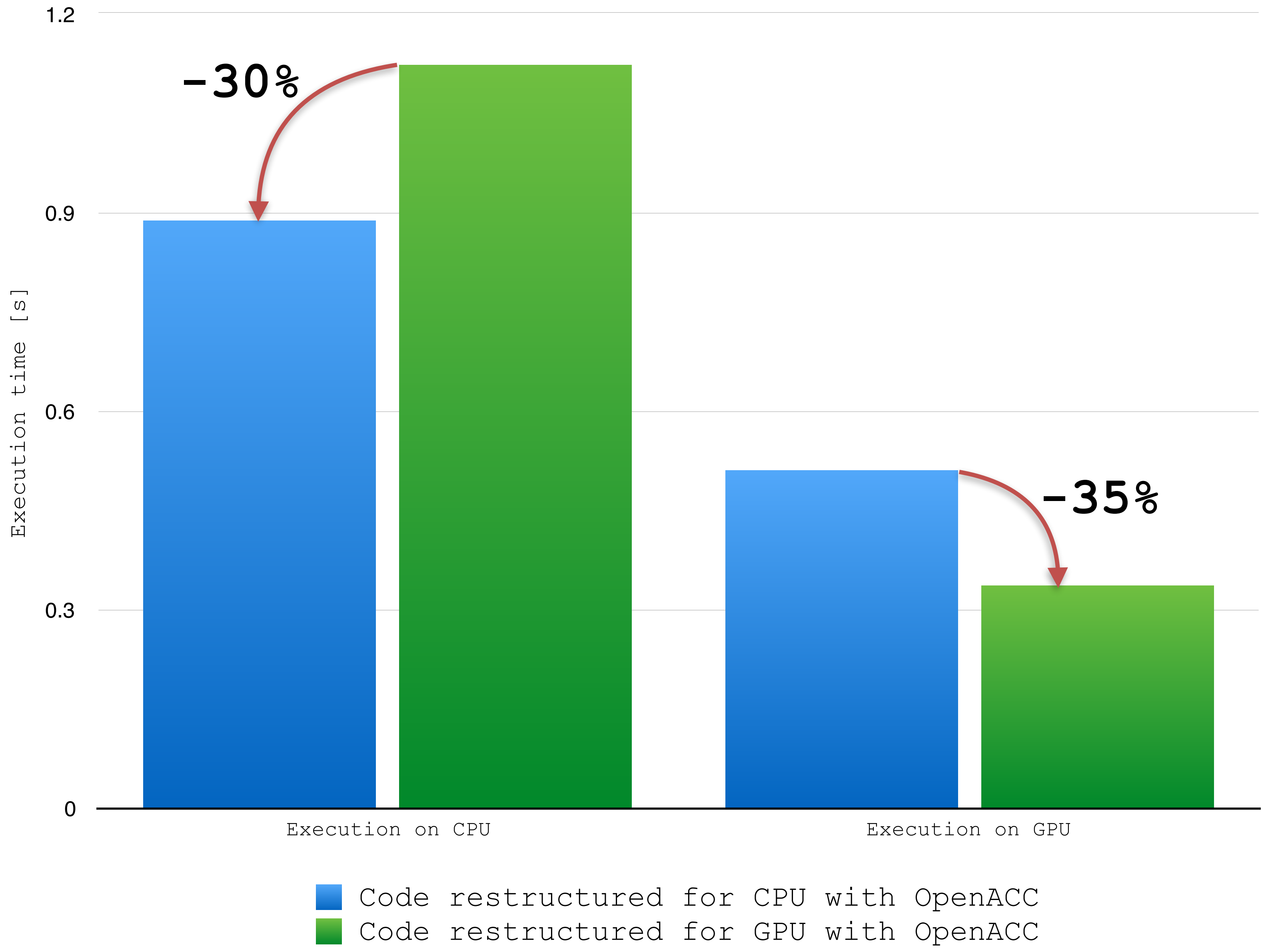
```
DO k=1,nz
  CALL fct()
  DO j=1,nproma
    ! 1st loop body
  END DO
  DO j=1,nproma
    ! 2nd loop body
  END DO
  DO j=1,nproma
    ! 3rd loop body
  END DO
END DO
```

CPU optimal

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

GPU optimal

Performance portability: COSMO Radiation



C2SM Center for Climate Systems Modeling

Code Maintenance Problem

```
#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
```

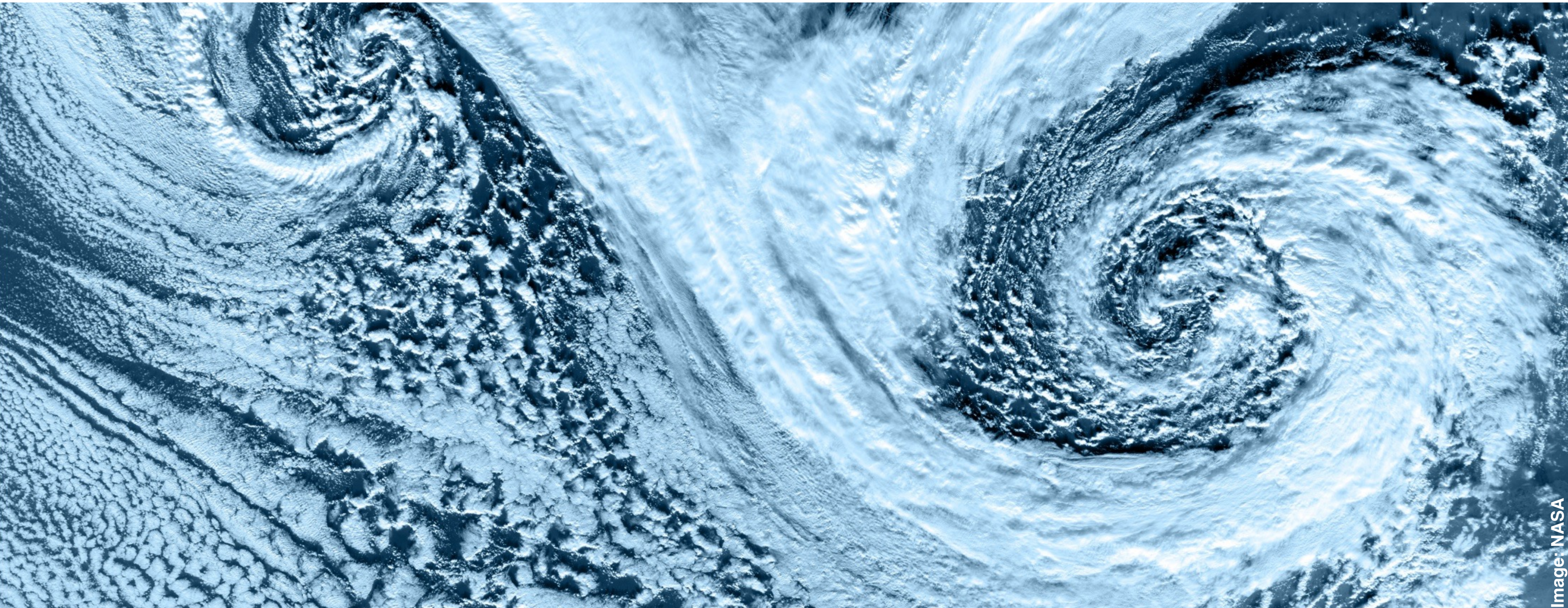
CPU

GPU

- Multiple code paths
- Hard maintenance
- Error prone
- Domain scientists have to know well each target architectures

C2SM Center for Climate Systems Modeling

CLAW low-level transformations



CLAW: Low-level transformations

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

clawfc →

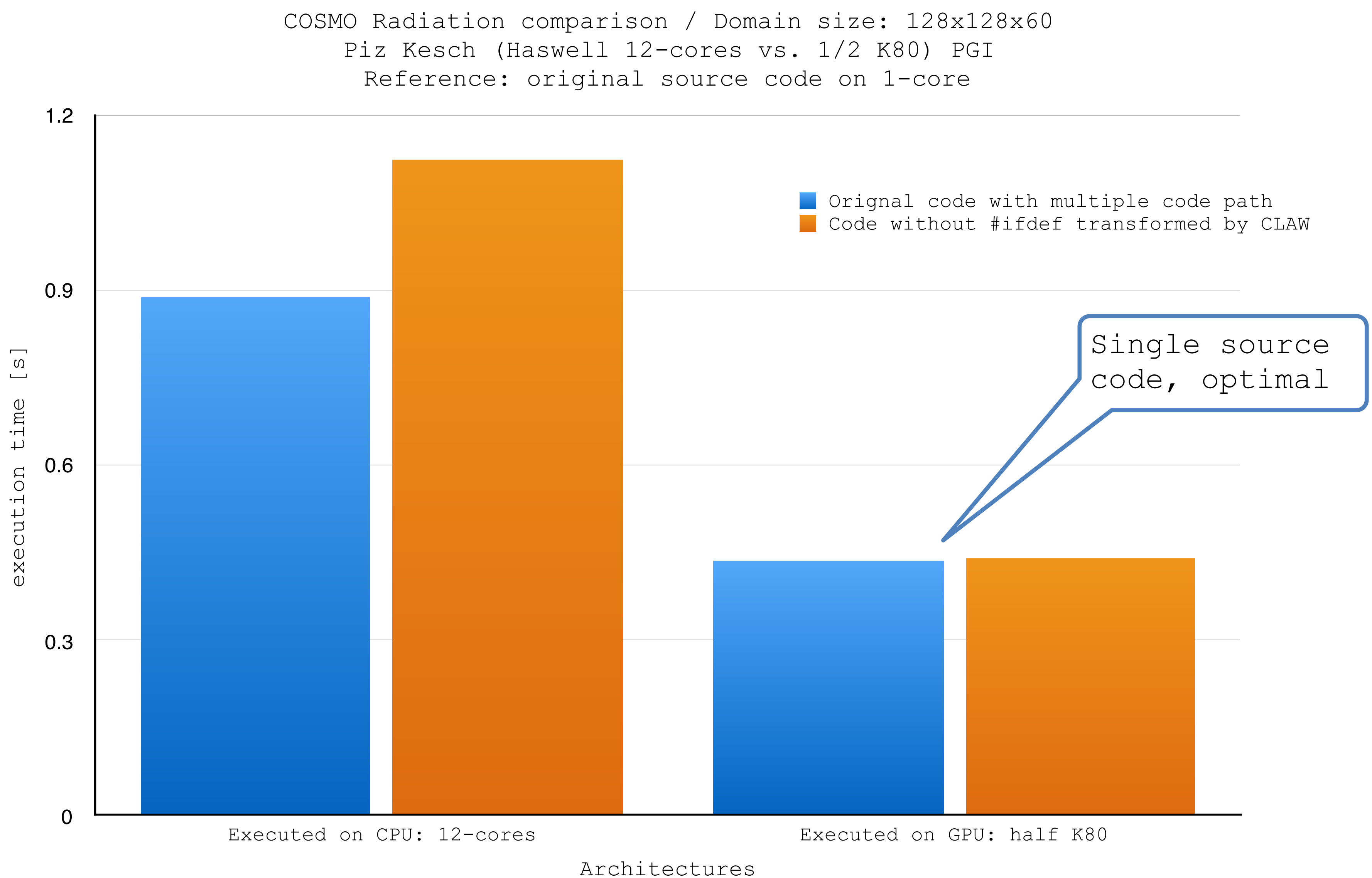
CPU to GPU

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

CLAW Compiler Low-Level Transformations:

- Loop fusion
- Loop reordering
- Loop extraction
- Loop hoisting
- Caching
- On the fly computation
- Array notation to do statement
- Code removal
- Conditional directive enabling

CLAW: Low-level transformations



Too many directives? Too complicated?

```
!$acc parallel loop
!$claw loop-interchange
DO k=1,nz
  !$claw loop-extract fusion
  CALL fct()
  !$claw loop-fusion
  !$acc loop
  !$omp parallel do
  DO j=1,nproma
    ! 1st loop body
  END DO
  !$omp end parallel do
  !$claw loop-fusion
  !$acc loop
  !$omp parallel do
  DO j=1,nproma
    ! 2nd loop body
  END DO
  !$omp end parallel do
  !$claw loop-fusion group(j)
  !$acc loop
  !$omp parallel do
  DO j=1,nproma
    ! 3rd loop body
  END DO
  !$omp end parallel do
END DO
!$acc end parallel
```

Typical code could includes the following compiler directives:

- OpenMP
- OpenACC
- CLAW
- …

Still targeted for performance aware developper. Doesn't help the domain scientist.

Can we do it in a simpler way?

C2SM
Center for Climate
Systems Modeling

CLAW high-level abstraction

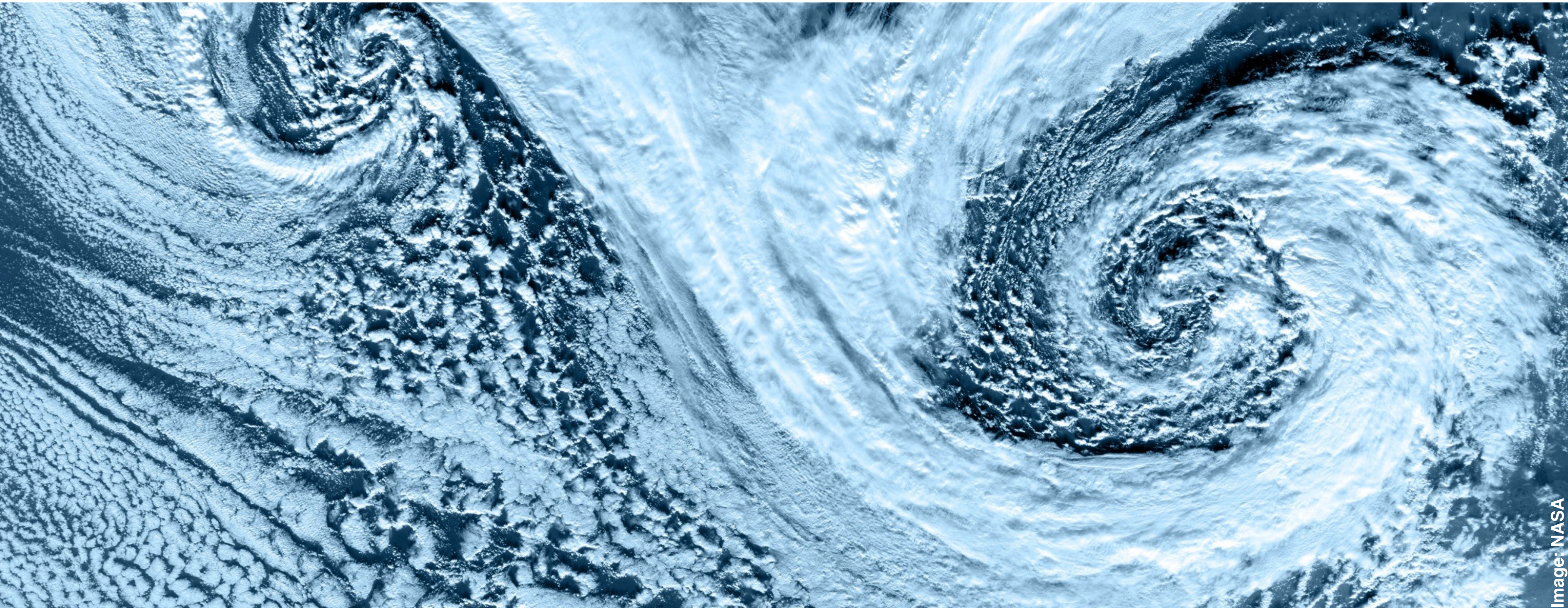
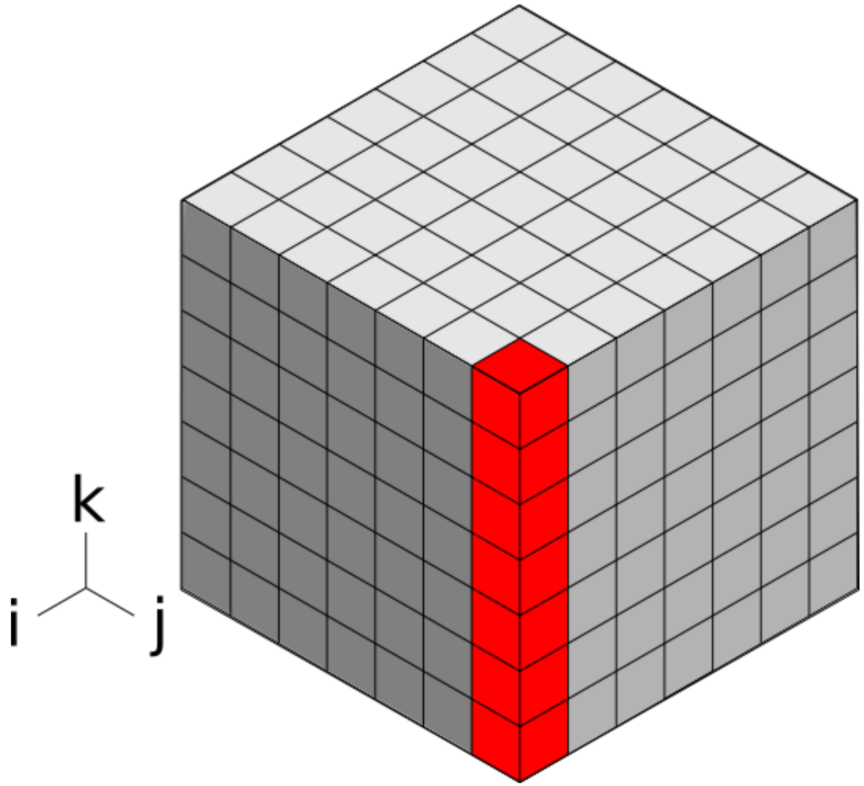


Image: NASA

CLAW One column abstraction



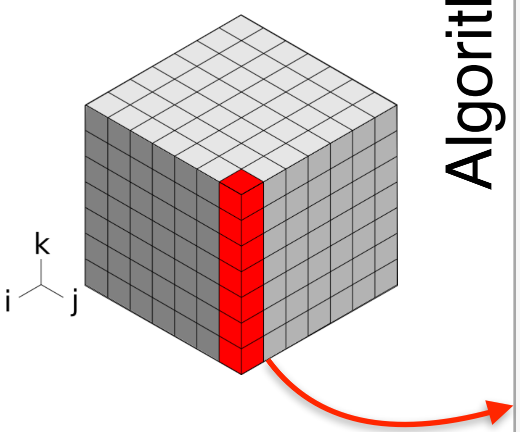
Separation of concerns

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

CLAW One column abstraction

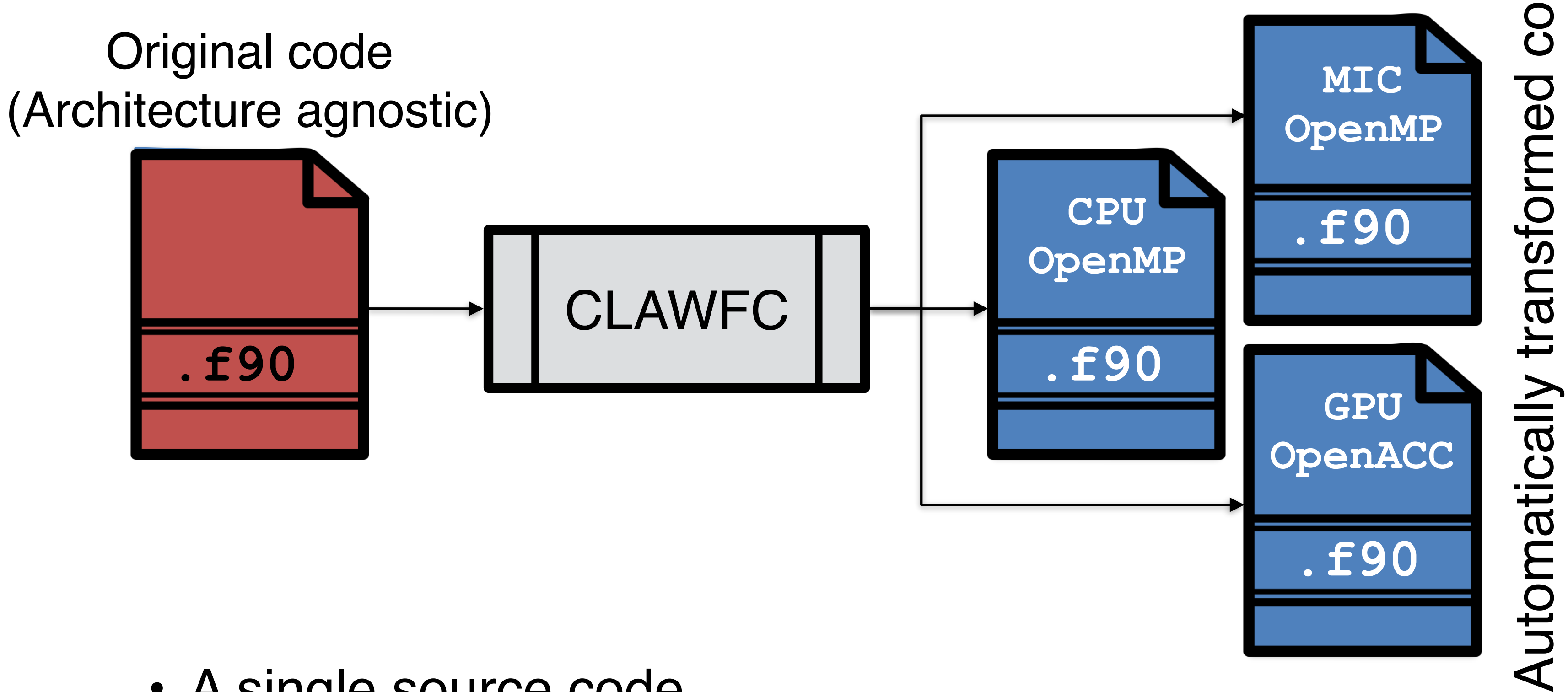
Algorithm for one column only

```
SUBROUTINE lw_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 lw_solver
```



Dependency on the vertical dimension only

CLAW One column abstraction



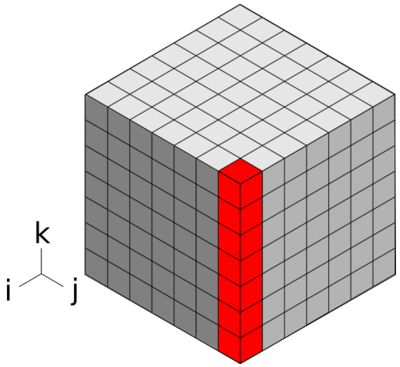
- A single source code
- Specify a target architecture for the transformation
- Specify a compiler directives language to be added

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

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

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

CLAW One column abstraction



Automatic promotions

- Inside the parallelized subroutine
- Along the call graph

Iteration over the horizontal dimensions

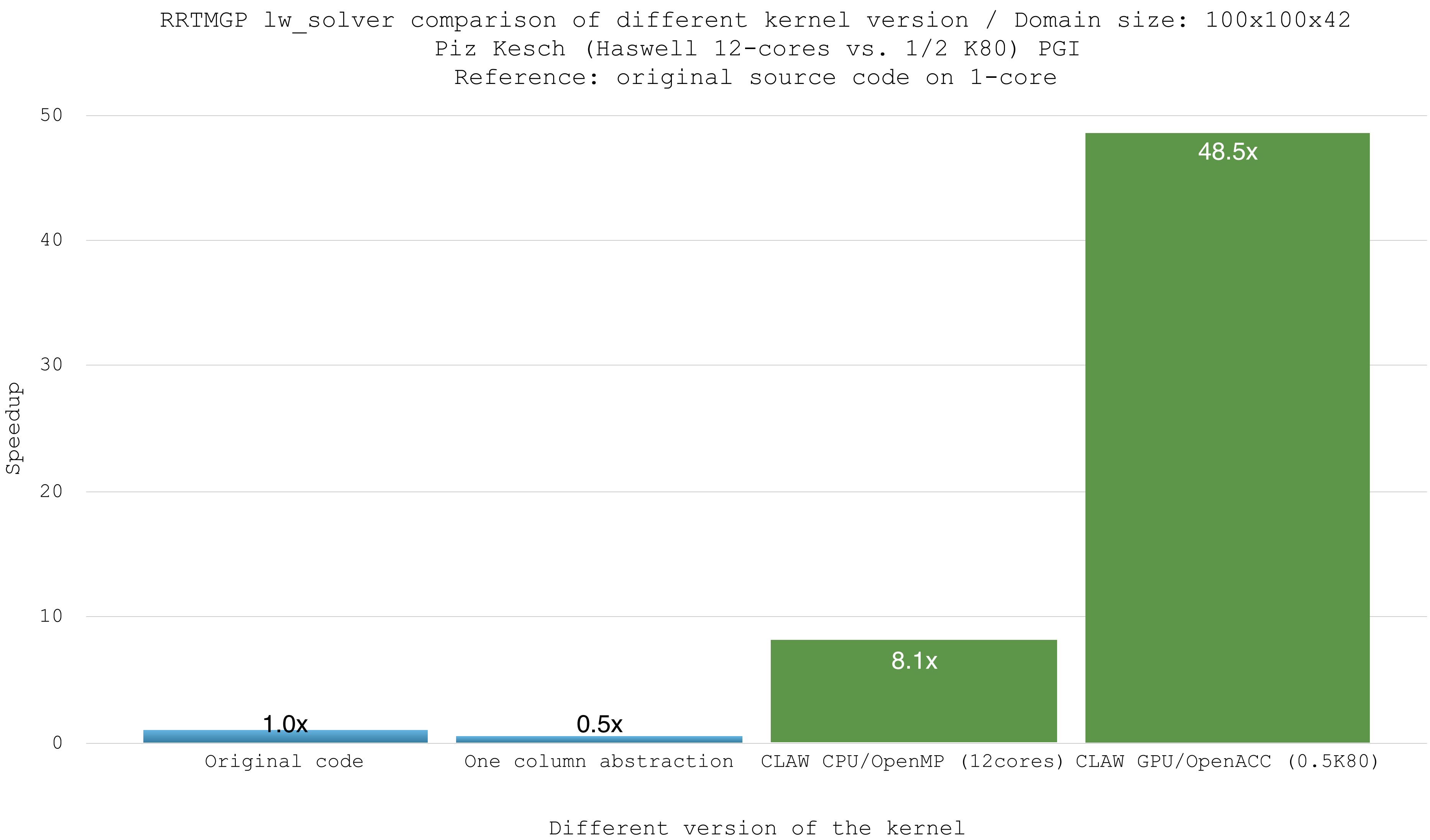
- Specific generation of do statements according to the architecture and data layout

Generation of directives

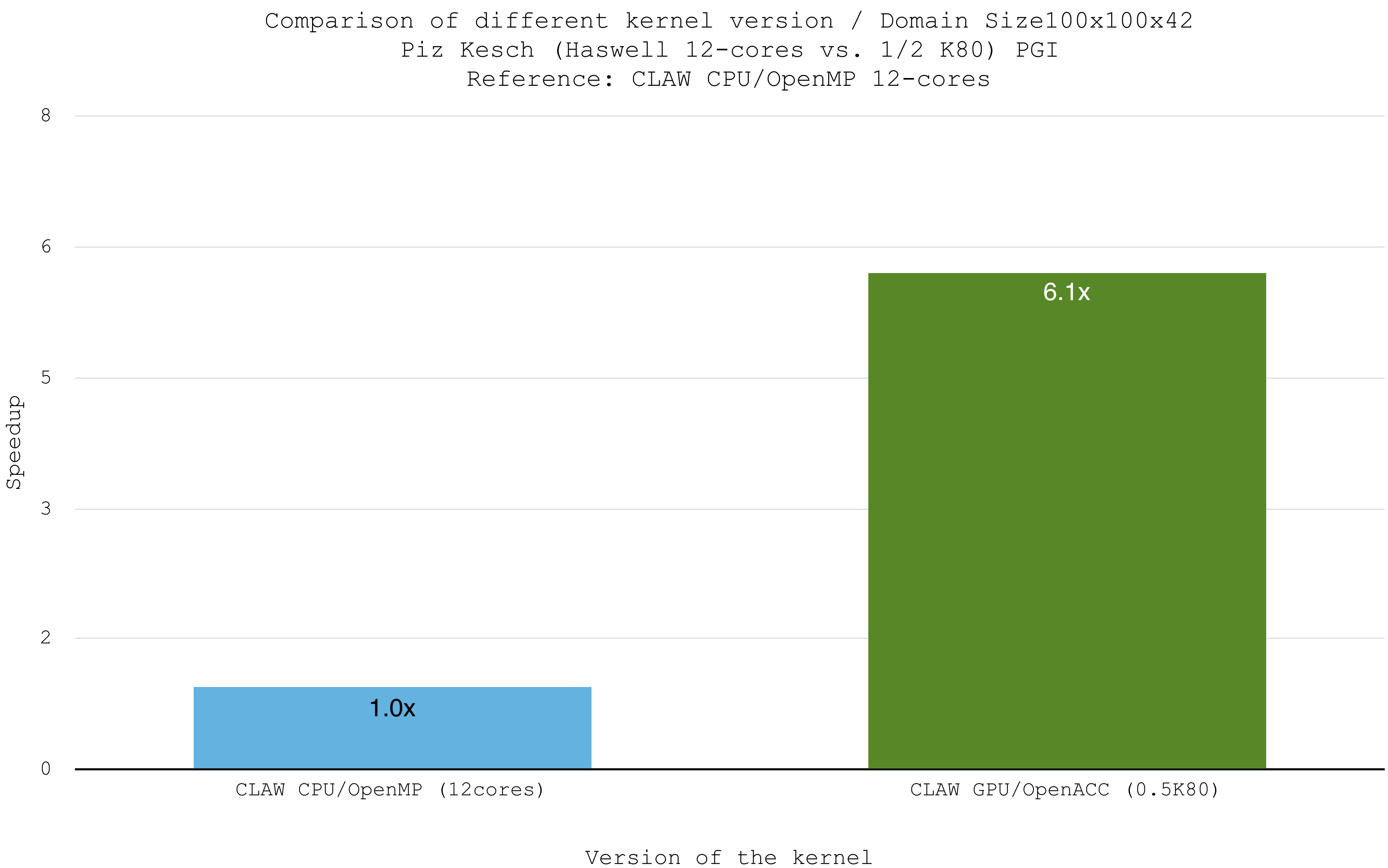
- OpenMP
- OpenACC

Currently, transformation rules are based on observation made in COSMO, HAMMOZ or ICON. Goal would be to add intelligence here or to couple it with tools that can drives the transformations

CLAW One column abstraction: early results

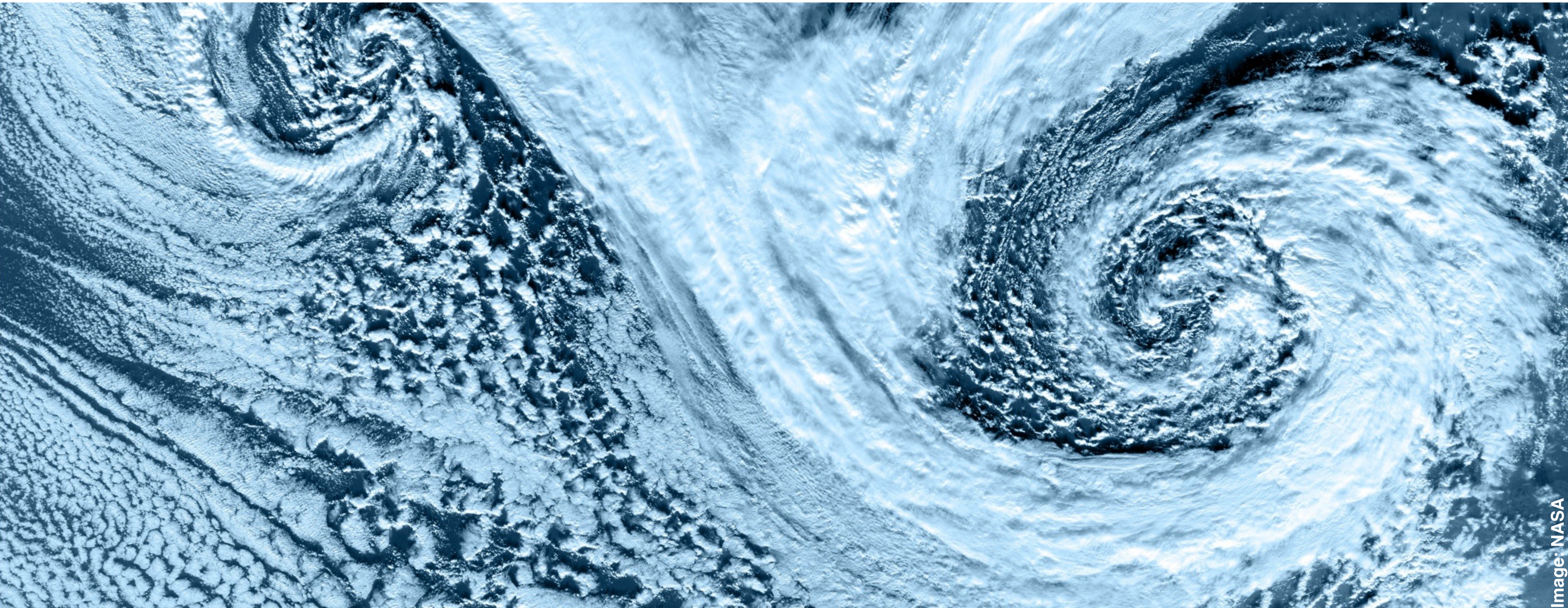


CLAW One column abstraction: early results



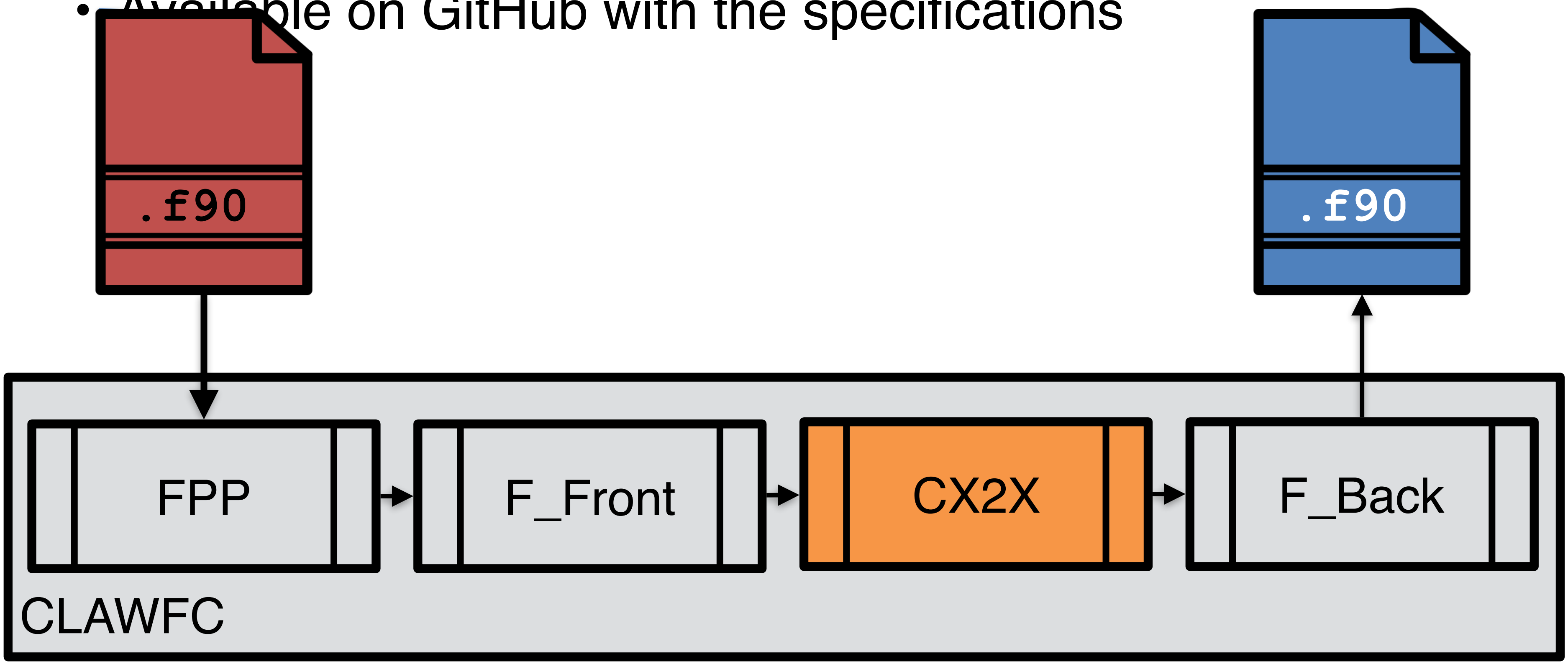
C2SM
Center for Climate
Systems Modeling

CLAWFC under the hood



CLAW source-to-source translator

- Based on the OMNI Compiler FORTRAN front-end & back-end
- Source-to-source translator
- Open source under the BSD license
- Available on GitHub with the specifications



CLAW XcodeML to XcodeML

```
SUBROUTINE my_kernel(…)
   !$claw define dimension icol(1:ncol) &
   !$claw parallelize

   DO k = 1, nz
      ! Column do stmt body
   END DO

END SUBROUTINE my_kernel
```

↓

```
<XcodeProgram>
  <FfunctionDefinition>
    <name>my_kernel</name>
    <body>
      <FpragmaStatement>claw define …</FpragmaStatement>
      <FdoStatement>
        <!-- loop body -->
      </FdoStatement>
    </body>
  </FfunctionDefinition>
</XcodeProgram>
```

CLAW XcodeML to XcodeML

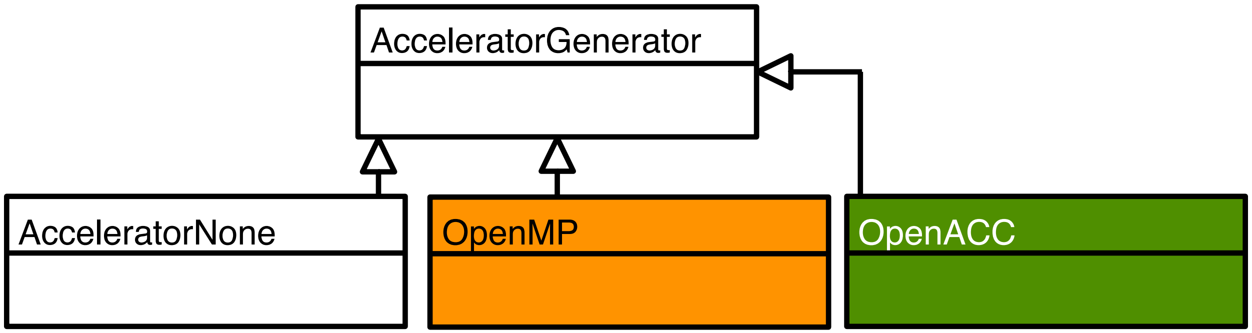
```
public class Parallelize extends Transformation
  public boolean analyze(){
    // Check if the transformation is possible
    return canBeTransformed;
  }

    public void transform(){
      if(_claw.getTarget() == Target.GPU){
        // Apply specific GPU transformation
      } else if(_claw.getTarget() == Target.CPU){
        // Apply specific CPU transformation
      } else if (_claw.getTarget == Target.MIC){
        // Apply specific MIC transformation
      }
    }
}
```

- Each transformation is an object with an analysis and a transformation step.
- Transformation objects have access to various information such as: target architecture, desired directives language …

CLAW XcodeML to XcodeML

Abstracted class representing the directive languages for easy generation



Java code

```
AcceleratorGenerator.genParallelRegion(startStmt, endStmt);
AcceleratorGenerator.genParallelLoop(doStmt);
```

XcodeML/F OpenMP

```
<FpragmaStatement>omp parallel</FpragmaStatement>
<FpragmaStatement>omp do</FpragmaStatement>
<FdoStatement><!-- loop body --></FdoStatement>
<FpragmaStatement>omp end do</FpragmaStatement>
<FpragmaStatement>omp end parallel</FpragmaStatement>
```

XcodeML/F OpenACC

```
<FpragmaStatement>acc parallel</FpragmaStatement>
<FpragmaStatement>acc loop</FpragmaStatement>
<FdoStatement><!-- loop body --></FdoStatement>
<FpragmaStatement>acc end parallel</FpragmaStatement>
```

CX2X: CLAW XcodeML to XcodeML

- Modular transformation framework based on OMNI tools
- Each directive triggers a transformation on an XcodeProgram
  - Single line directive
  - Block directive
- Order of application of transformations is configurable
  - Transformation may or may not have an impact on each other
- Transformation have several information to choose how to modify the original code
  - Target architecture: `cpu/gpu/mic` …
  - Directive language: `none/openmp/openacc` …
  - Internal analysis based on XcodeML/F
  - Target compiler?

CLAW Current status and next steps

- Low-level transformations implemented
  - Used in current radiation code of COSMO
  - Several mini-app extracted from HAMMOZ
- High-level abstraction under investigation with RRTMGP code from ICON model
  - Helps to specify the CLAW directives and clauses needed for such abstraction
  - Could draw some limitation on what can be used in a “parallelized” subroutine

Resources

https://github.com/C2SM-RCM

claw-compiler

CLAW Fortran Compiler

Updated an hour ago

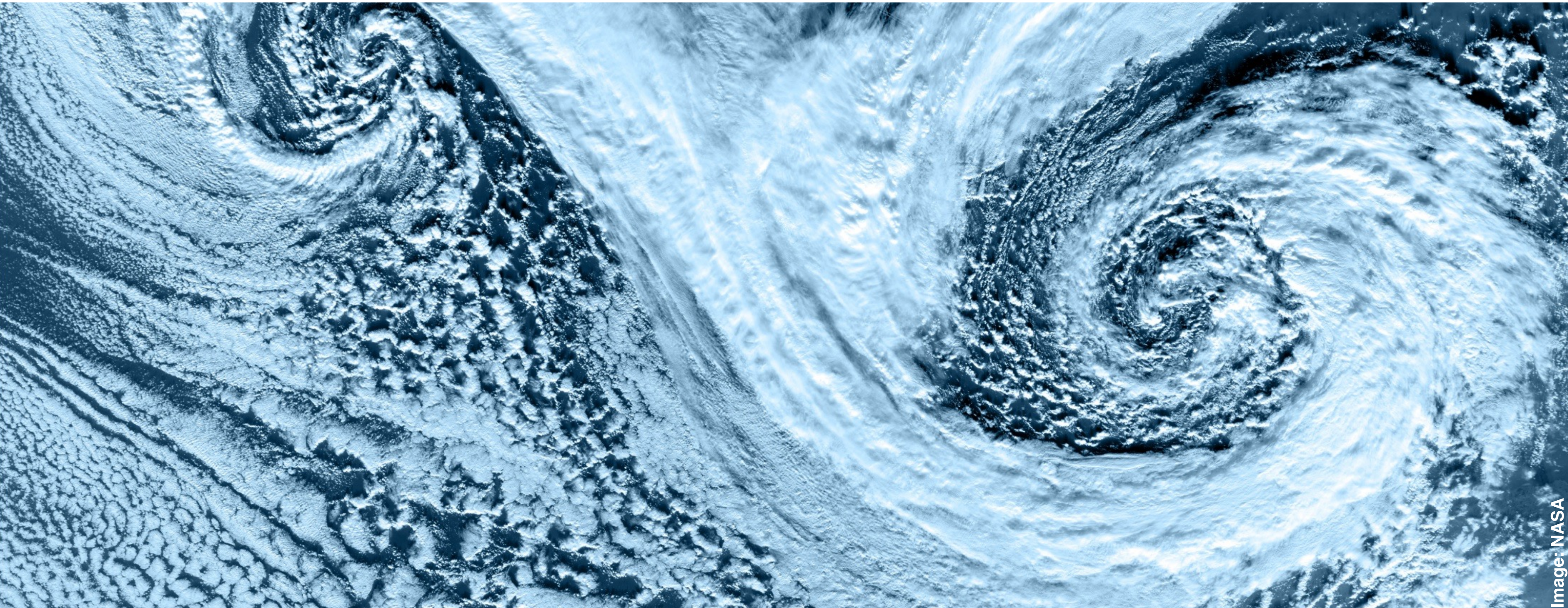
claw-language-specification

Specification of the CLAW language

Updated 2 days ago

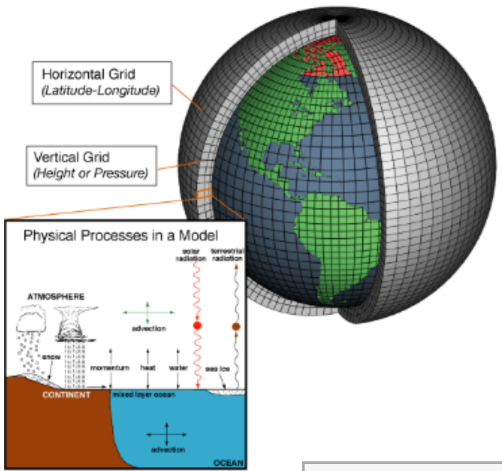
C2SM
Center for Climate
Systems Modeling

Other initiative at MeteoSwiss

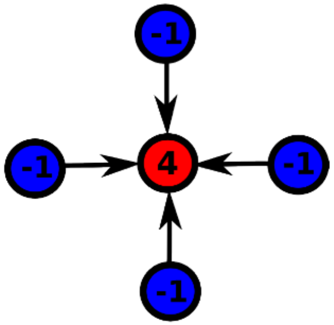


GridTools (performance portability C++)

Set of tools for solving PDEs on multiple grids.



Laplace: ?^{2}



lap(i,j) = -4 * u(i,j) +
u(i+1,j) + u(i-1,j) +
u(i,j-1) + u(i,j+1)

```
struct Laplace{
  typedef in_accessor<0, extent<-1,1,-1,1> > u;
  typedef out_accessor<1> lap;

  template <typename Evaluation>
  static void Do(Evaluation const& eval, full_domain){
    eval(lap()) = eval(-4*u() + u(i+1) + u(i-1) +
                           u(j+1) + u(j-1));
  }
};
```

GridTools (performance portability C++)

Stencil DSL

Boundar y

Halo exchanges

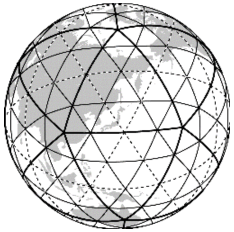
Storages

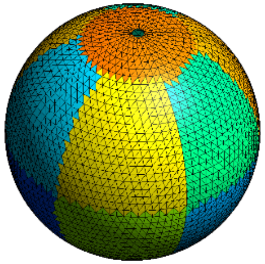
Grids

Rectangular

Icosahedral /

Cubed





architecture specific implementation

GPU (CUDA)

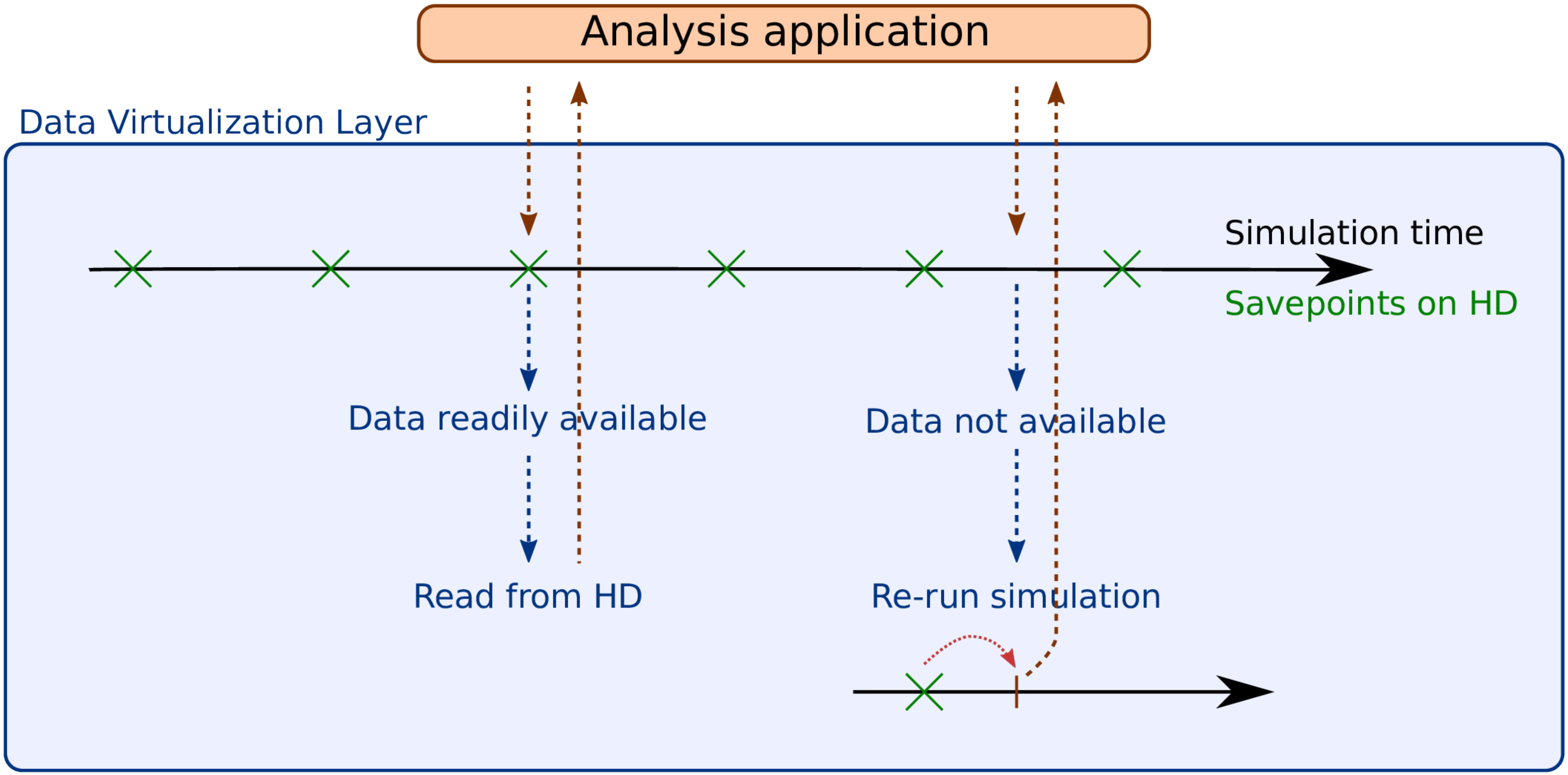
x86

Xeon Phi

C++ template meta-programming

carlos.osuna@meteoswiss.ch

crClim (Portable bit reproducibility)



crClim (Portable bit reproducibility)

Shadowing

```fortran
MODULE radiation

USE trigo

! some code and computation

y = TAN(x)

! some code and computation
END MODULE radiation
```

```fortran
MODULE trigo
  use iso_fortran_env

interface TAN
    module procedure MCH_TAN
  end interface TAN

contains
  real(kind=real64) function MCH_TAN(x) result(y)
    !$acc routine seq
    real(kind=real64), intent(in) :: x
    ! computation tan(x) stored in y
  end function MCH_TAN(x)

END MODULE trigo
```

crClim (Portable bit reproducibility)

- The exponentiation in Fortran an intrinsic operator

```
z = x**y
```

- Unable to override the behavior of an intrinsic operator for the native type (i.e. real, integer, …)
- Need to replace the usage of ** by a user defined function

```
pure real function pow(x, y) result(z)
```

- The exponentiation is automatically replaced by a custom preprocessor

christophe.charpilloz@meteoswiss.ch

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

C2SM Center for Climate Systems Modeling

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

valentin.clement@env.ethz.ch

Persons involved in the CLAW project:
Jon Rood (C2SM), Sylvaine Ferrachat (ETH Zurich), Will Sawyer (CSCS), Oliver Fuhrer (MeteoSwiss), Xavier Lapillonne (MeteoSwiss)