Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Home Affairs FDHA
Federal Office of Meteorology and Climatology MeteoSwiss

Operational numerical weather prediction on a GPU-accelerated cluster supercomputer

C. Osuna, O. Fuhrer, X. Lapillonne, V. Clement, P. Spoerri, A. Walser, A. Arteaga, T. Gysi, S. Ruedisuehli, K. Osterried, T. Schulthess

C2SM
Center for Climate
Systems Modeling

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

OLD Meteoswiss operational system

Cray XE6 (Albis/Lema)

MeteoSwiss operational system

~4 years

OLD Meteoswiss operational system

ECMWF-Model

16 km gridspacing

2 x per day 10 day forecast

COSMO-7

Δx = 6.6 km, Δt = 60 s

393 x 338 x 60 cells

3 x per day 72 h forecast

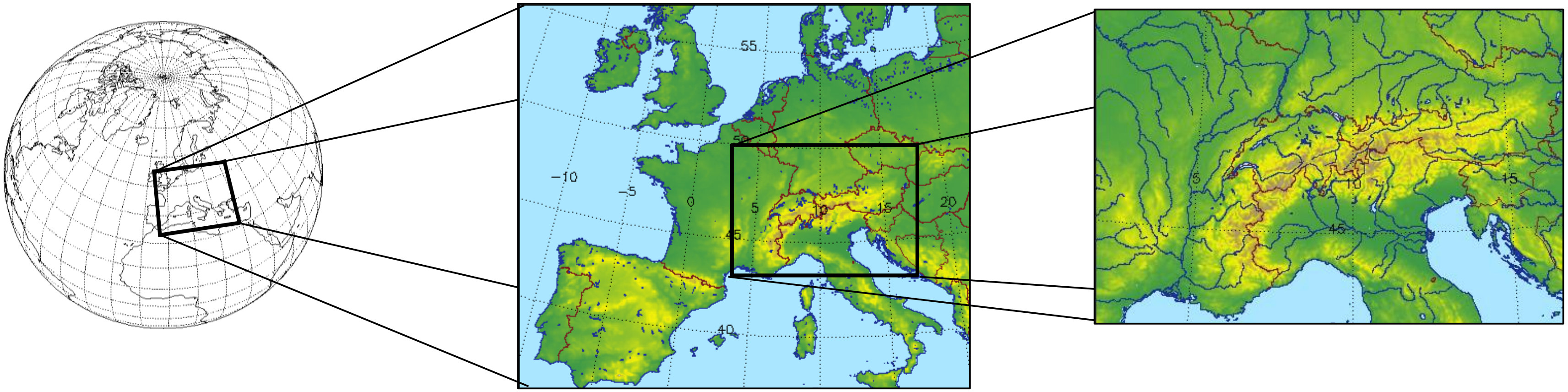
COSMO-2

Δx = 2.2 km, Δt = 20 s

520 x 350 x 60 cells

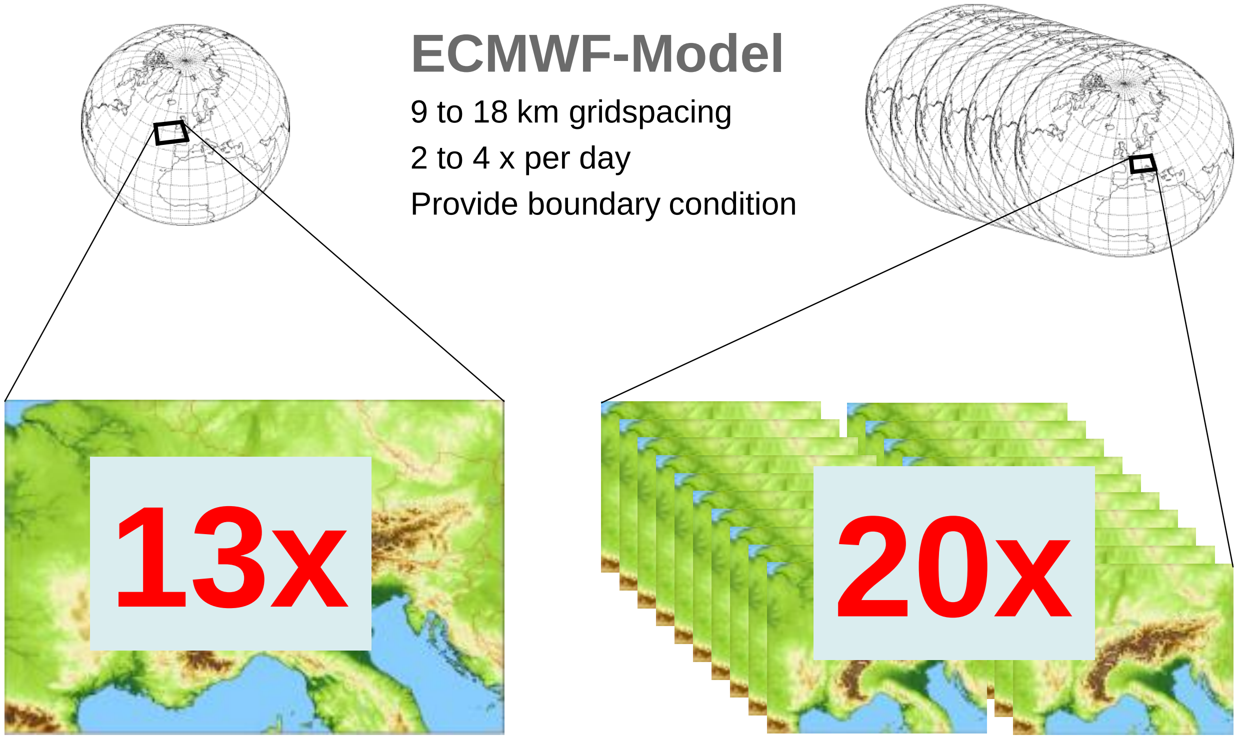
7 x per day 33 h forecast

1 x per day 45 h forecast



Requirements for New operational setup

- COSMO-1 (1 km high resolution) and COSMO-E (ensemble)
- Total computation cost for requirements = **40x**



ECMWF-Model

9 to 18 km gridspacing
2 to 4 x per day
Provide boundary condition

COSMO-1

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

(in production since
1/04/2016)

13x

20x

COSMO-E

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

(production planned
Mai/June 2016)

Ensemble **ation: LETKF**

7x

How to reach 40x in 5 years?

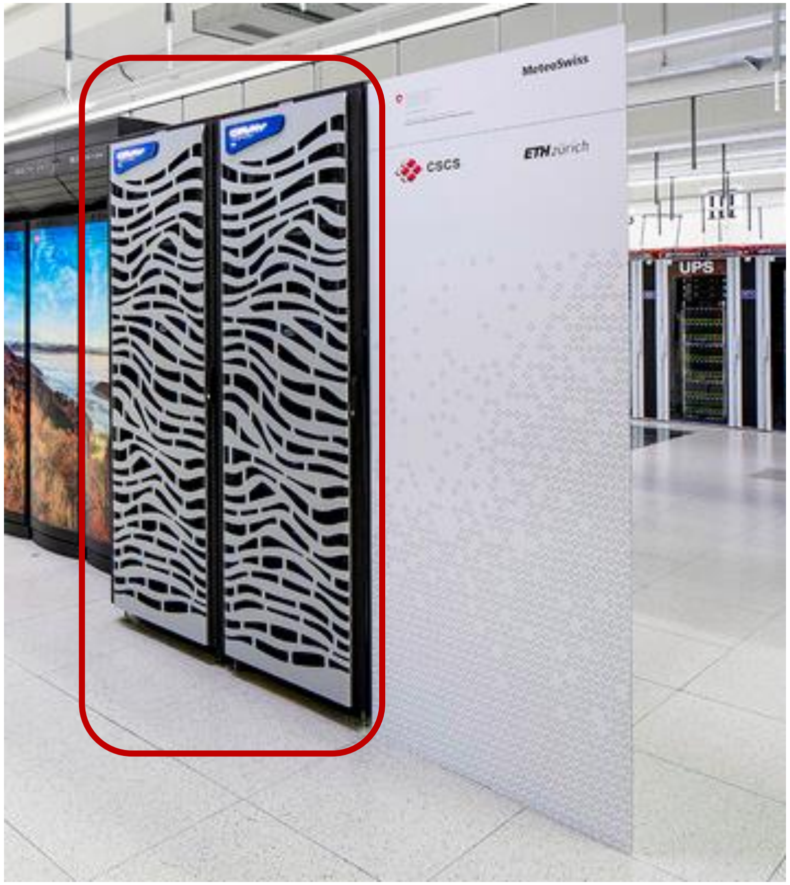
Key Ingredients

Processor performance (Moore’s Law)	~2.8x	
Increase in number of sockets	~1.3x	
Port to accelerators (GPUs)	~2.3x	~4x
Code improvement	~1.7x	
Increase utilization of system	~2.8x	

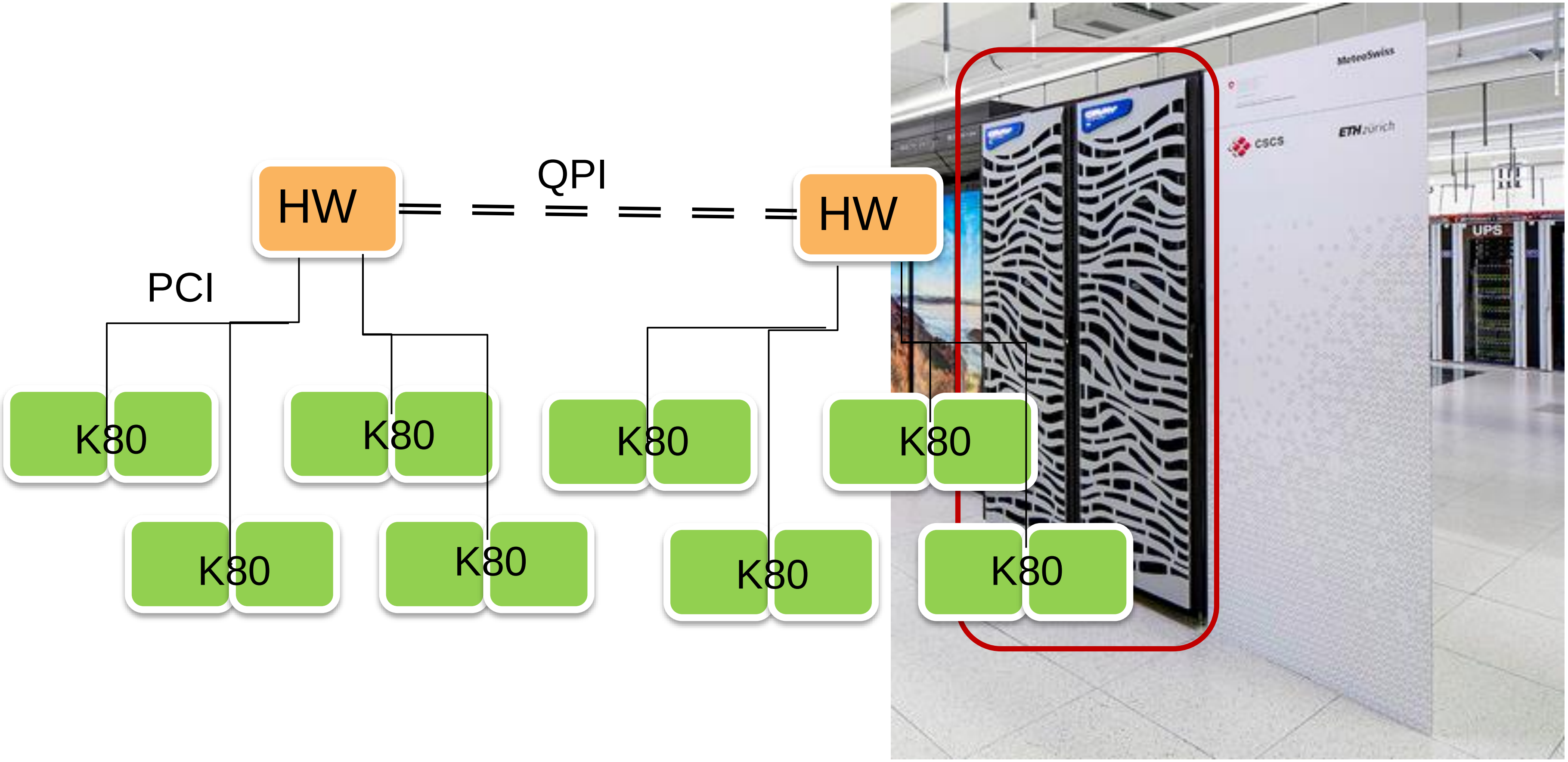
New MeteoSwiss hybrid HPC system

Piz Kesch (Cray CS Storm)

- 2 Cabinets (production & failover) installed at CSCS in July 2015
- 12 “Fat” compute nodes per cabinet
  - 2 Intel CPU Xeon E5 2690 (Haswell)
  - **8** Tesla K80 GPUs (each with 2 GK210 chip)

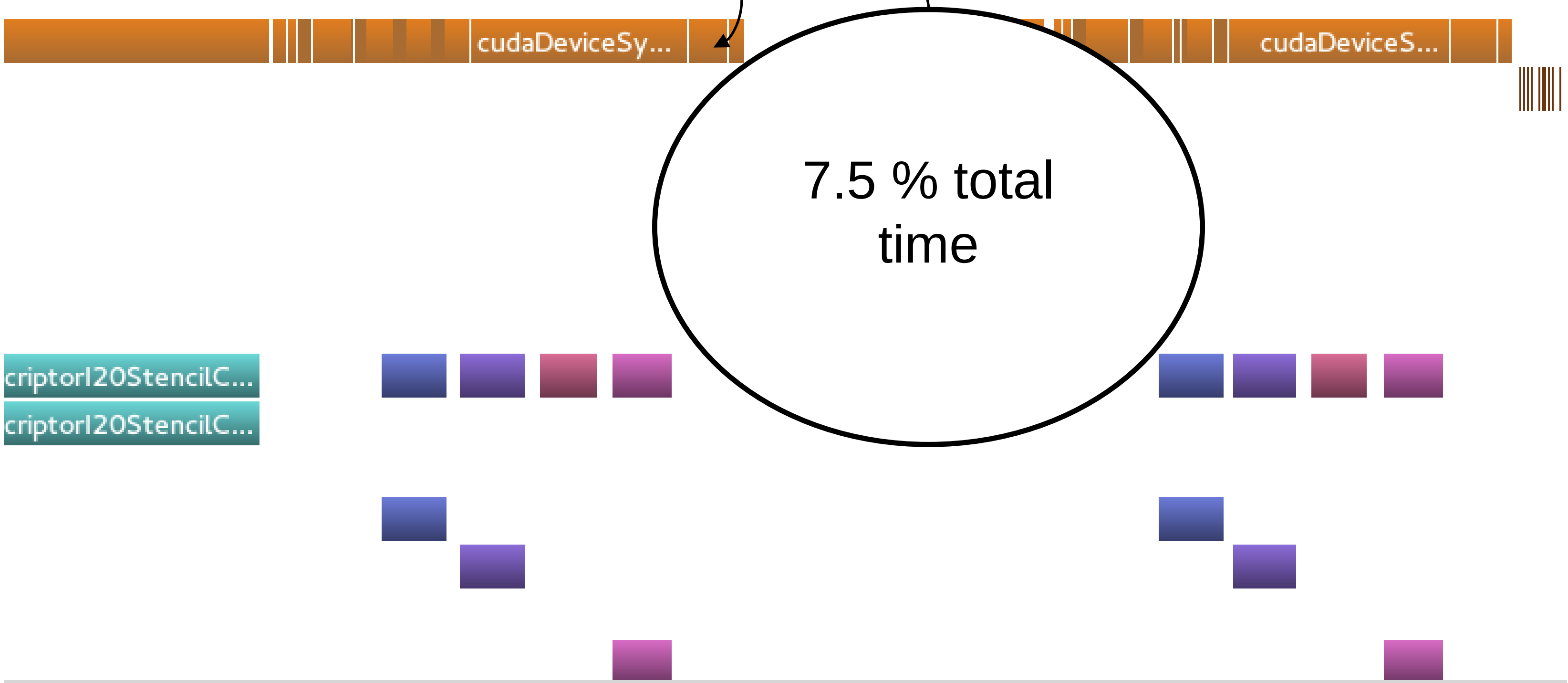


New MeteoSwiss hybrid HPC system



GPU communication with MPI

- Every Pack-MEvery pair GCL::Pack-MPI_Isend is generating large gaps in the streams due to delays in the host timeline (cudaDeviceSynchronize and cuda API calls)
- Collaboration with OSU, we will need mvapich2 GDS



cudaEventSynchronize

cudacudaEventElapsedTime
EventRecord

7.5 % total time

How to port a full weather model to GPUs?

Up or down?

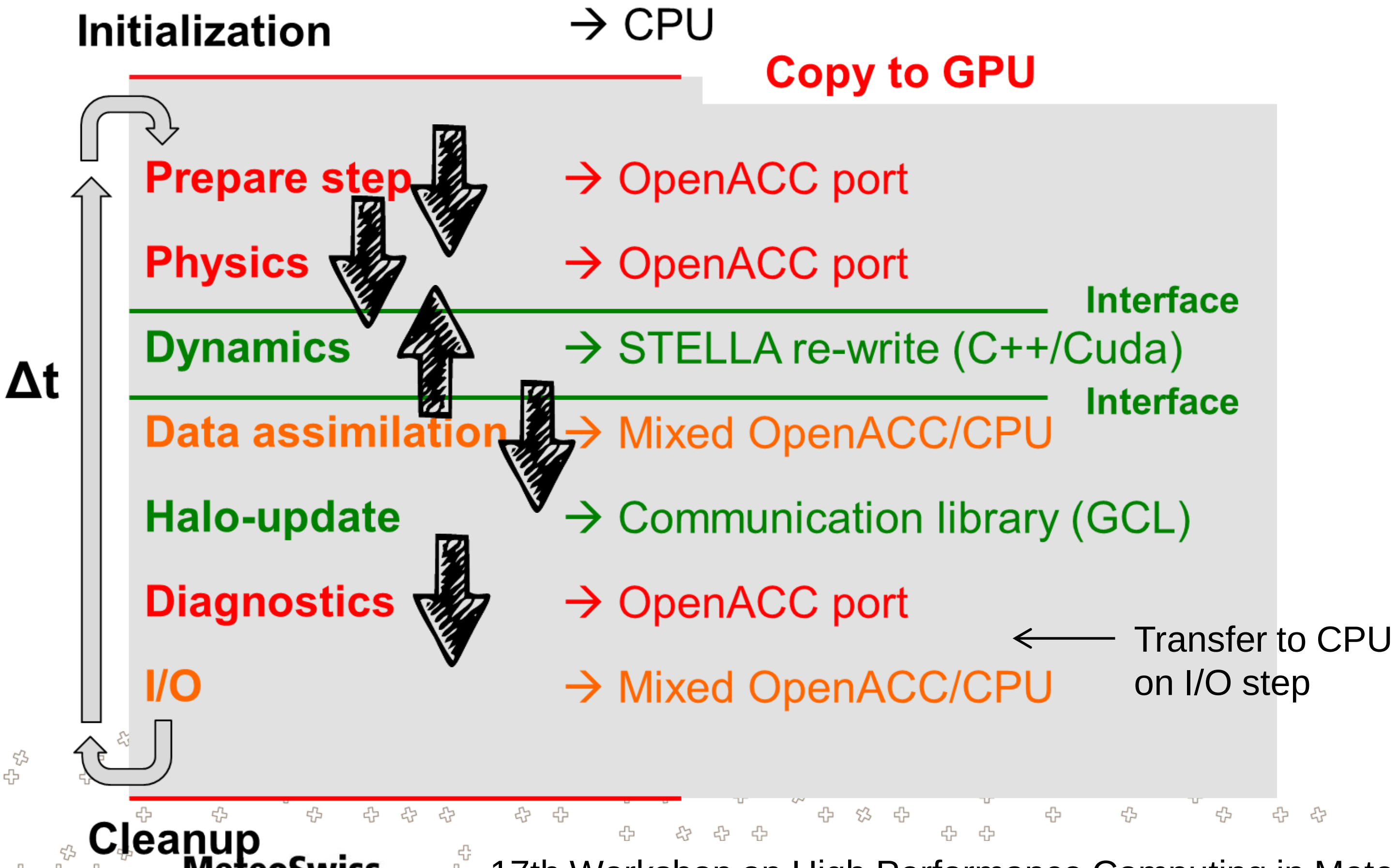
- **Increase level of abstraction**
  - Remove details of implementation
  - “Disruptive change”

- **Lower level of abstraction**
  - Add implementation details
  - „Incremental change“

Source: Oliver Fuhrer

The COSMO model on GPU

- Take advantage of the high computational capacity of GPUs
- Low compute intensity : avoid GPU-CPU data transfer
- Full GPU port strategy : all computations on the GPU



Initialization → CPU

Copy to GPU

Δt

Prepare step → OpenACC port

Physics → OpenACC port

Interface

Dynamics → STELLA re-write (C++/Cuda)

Interface

Data assimilation → Mixed OpenACC/CPU

Halo-update → Communication library (GCL)

Diagnostics → OpenACC port

I/O → Mixed OpenACC/CPU

← Transfer to CPU on I/O step

Cleanup

DOWN – Lower Level of abstraction

- Incremental adaptation to various programming models
- Traditionally OpenMP, OpenACC,
- Limited flexibility to custom platform dependent adaptations (#ifdef)

```
DO jb = i_startblk, i_endblk
    DO jc = i_startidx, i_endidx
        DO jk = slev, elev
            div_vec_c(jc,jk,jb) = vec_e(iid(ic,jb,1),jk,iblk(ic,jb,1)) * ptr_int%geofac_div(jc,1,jb) + &
                      vec_e(iid(ic,jb,2),jk,iblk(ic,jb,2)) * ptr_int%geofac_div(jc,2,jb) + &
                      vec_e(iid(ic,jb,3),jk,iblk(ic,jb,3)) * ptr_int%geofac_div(jc,3,jb)
        ENDDO
    ENDDO
ENDDO
```

DOWN – Lower Level of abstraction

- Incremental adaptation to various programming models
- Traditionally OpenMP, OpenACC,
- Limited flexibility to custom platform dependent adaptations (#ifdef)

```
!$ACC DATA COPYIN(vec_e) COPYOUT(div_vec_c)
!$ACC KERNELS LOOP, GANG(32), WORKER(8)
!OMP PARALLEL DO STATIC
                                             bl
                                             r_
                                             r_
     ENDDO
ENDDO
```

CPU

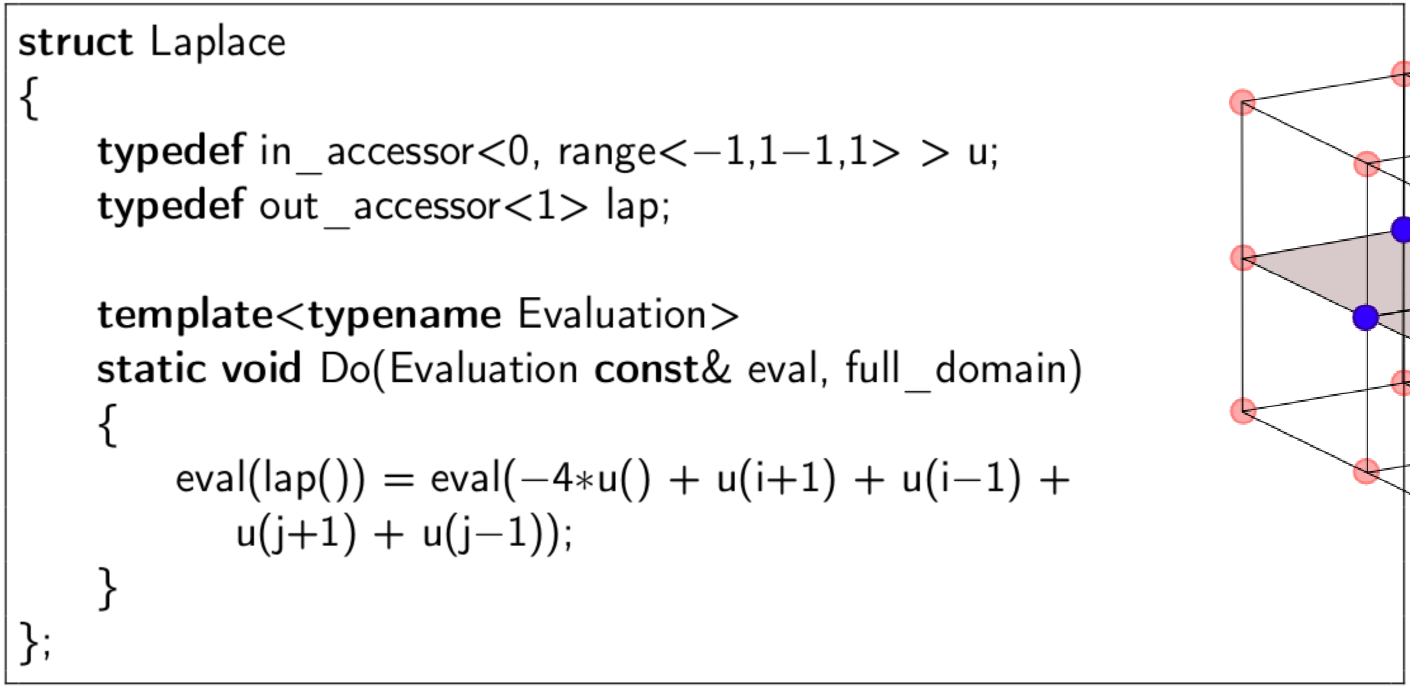
Algorithm: divergence
Language: Fortran
Grid: Structured
Data Layout: k, cell index, block index
Parallelism: block parallelism
Loop ordering: block index, cell index, k
Directives: OpenACC

GPU

Algorithm: divergence
Language: Fortran
Grid: Structured
Data Layout: cell index, k, block index
Parallelism: cell index
Loop ordering: block index, cell index, k
Directives: OpenACC

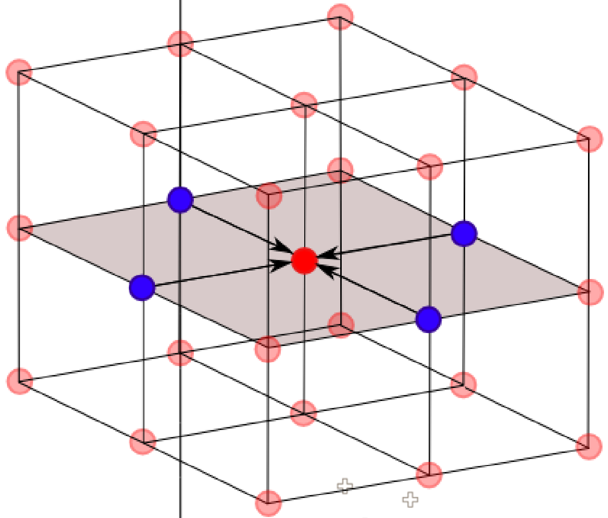
UP - STELLA / GridTools Library

- Domain-specific embedded language in C++
- Developed for stencil computations
- Separation of concern between model and hardware implementation
- Back-end optimized for different architectures (CPU,GPU)

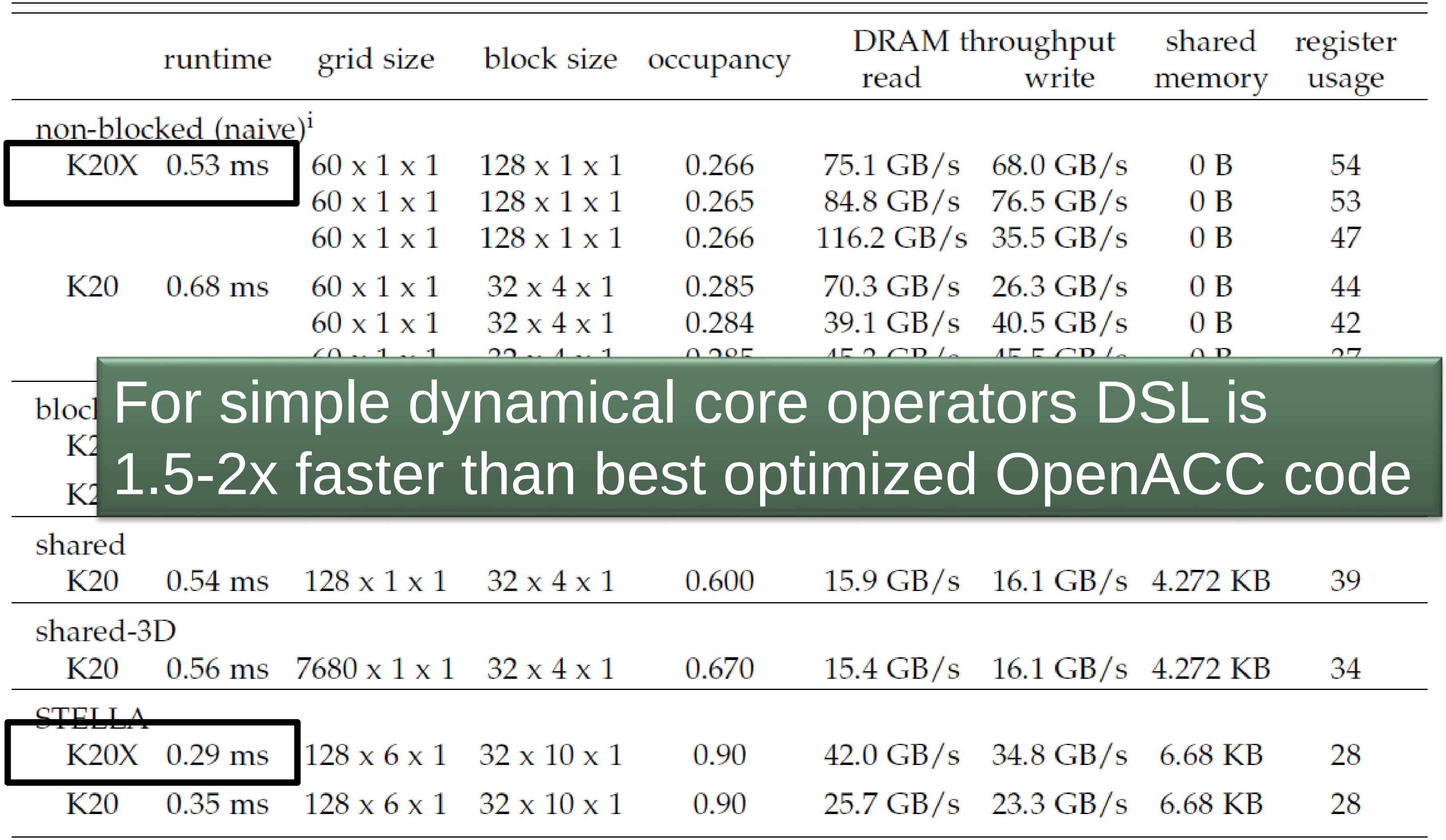


```
struct Laplace
{
    typedef in_accessor<0, range<-1,1-1,1> > u;
    typedef out_accessor<1> lap;

    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));
    }
};
```

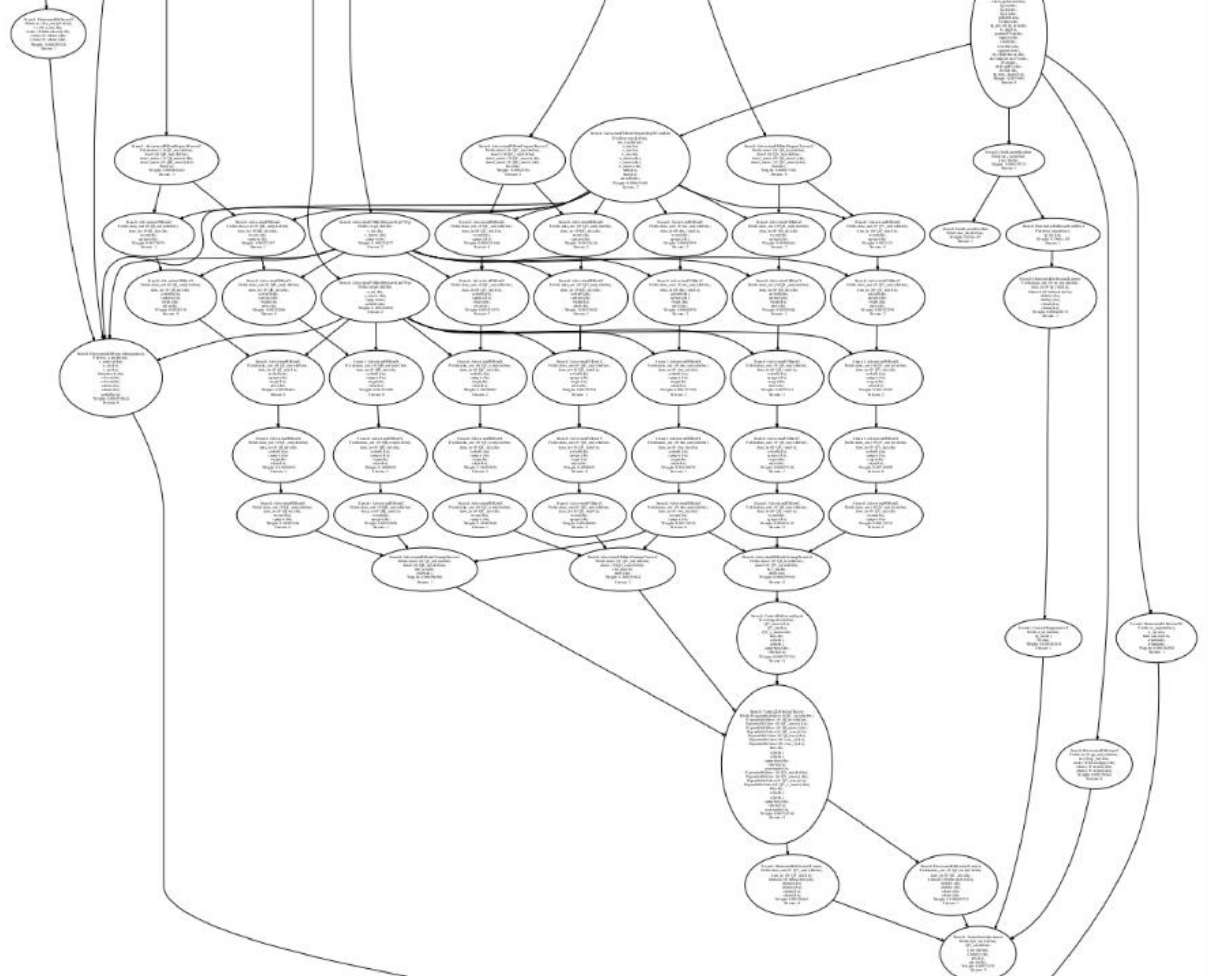


Performance Portability - Why UP (I)?



	runtime	grid size	block size	occupancy	DRAM throughput read	DRAM throughput write	shared memory	register usage
non-blocked (naive)[i]								
K20X	0.53 ms	60 x 1 x 1	128 x 1 x 1	0.266	75.1 GB/s	68.0 GB/s	0 B	54
		60 x 1 x 1	128 x 1 x 1	0.265	84.8 GB/s	76.5 GB/s	0 B	53
		60 x 1 x 1	128 x 1 x 1	0.266	116.2 GB/s	35.5 GB/s	0 B	47
K20	0.68 ms	60 x 1 x 1	32 x 4 x 1	0.285	70.3 GB/s	26.3 GB/s	0 B	44
		60 x 1 x 1	32 x 4 x 1	0.284	39.1 GB/s	40.5 GB/s	0 B	42
		60 x 1 x 1	32 x 4 x 1	0.285	45.3 GB/s	45.5 GB/s	0 B	37
bloc								
K2								
K2								
shared								
K20	0.54 ms	128 x 1 x 1	32 x 4 x 1	0.600	15.9 GB/s	16.1 GB/s	4.272 KB	39
shared-3D								
K20	0.56 ms	7680 x 1 x 1	32 x 4 x 1	0.670	15.4 GB/s	16.1 GB/s	4.272 KB	34
STELLA								
K20X	0.29 ms	128 x 6 x 1	32 x 10 x 1	0.90	42.0 GB/s	34.8 GB/s	6.68 KB	28
K20	0.35 ms	128 x 6 x 1	32 x 10 x 1	0.90	25.7 GB/s	23.3 GB/s	6.68 KB	28

For simple dynamical core operators DSL is 1.5-2x faster than best optimized OpenACC code



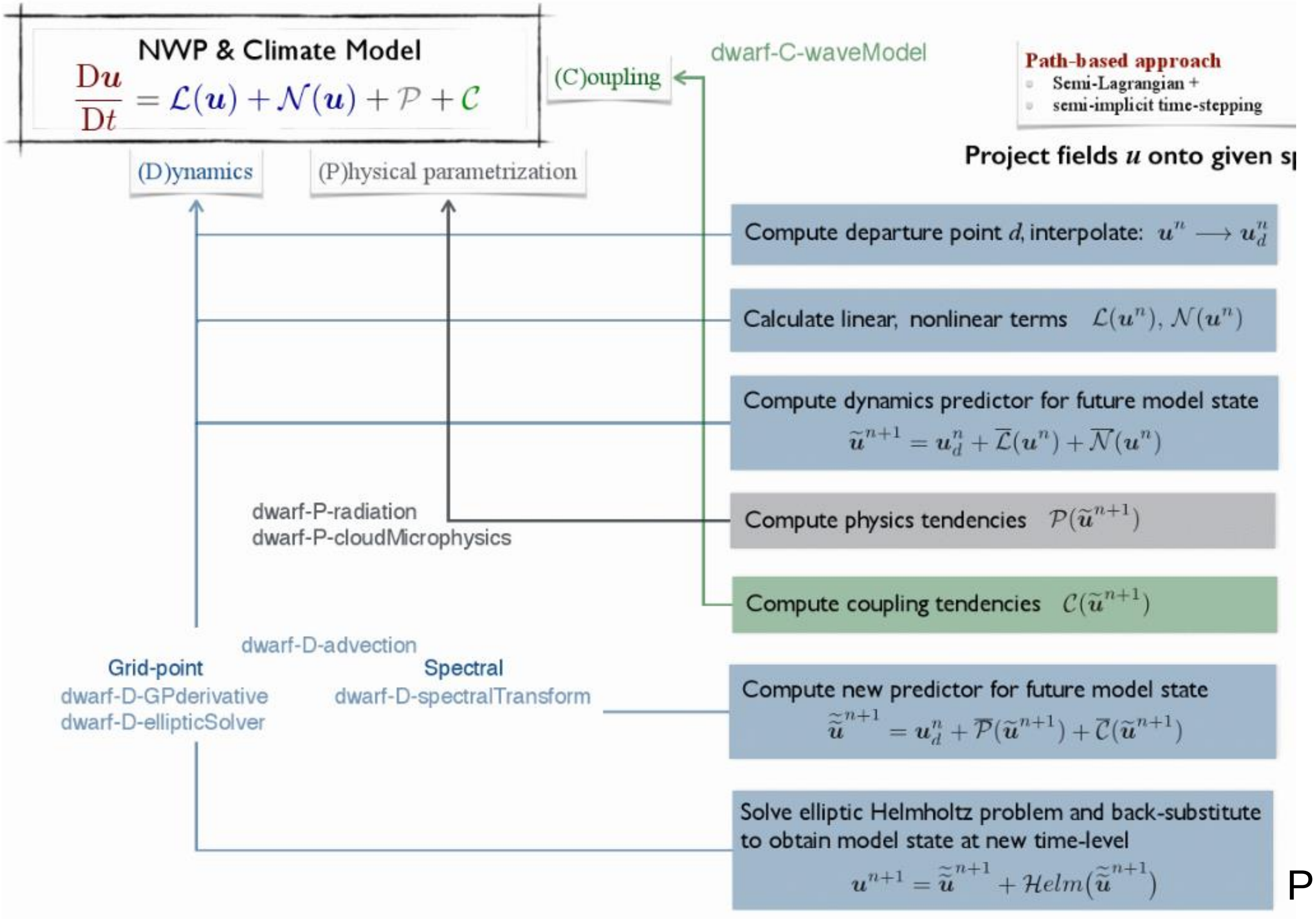
Funded by the
European Union

Preparing for performance portability of future models

ESCAPE

Funded by the
European Union

Energy efficient **SC**alable **A**lgorithms for weather **P**rediction at **E**xascale



NWP & Climate Model

$$\frac{D\boldsymbol{u}}{Dt} = \mathcal{L}(\boldsymbol{u}) + \mathcal{N}(\boldsymbol{u}) + \mathcal{P} + \mathcal{C}$$

(C)oupling

dwarf-C-waveModel

(D)ynamics

(P)hysical parametrization

Path-based approach
- Semi-Lagrangian +
- semi-implicit time-stepping

Project fields $\boldsymbol{u}$ onto given s

Compute departure point d, interpolate: $\boldsymbol{u}^n \longrightarrow \boldsymbol{u}^n_d$

Calculate linear, nonlinear terms $\mathcal{L}(\boldsymbol{u}^n), \mathcal{N}(\boldsymbol{u}^n)$

Compute dynamics predictor for future model state

$$\tilde{\boldsymbol{u}}^{n+1} = \boldsymbol{u}^n_d + \overline{\mathcal{L}}(\boldsymbol{u}^n) + \overline{\mathcal{N}}(\boldsymbol{u}^n)$$

dwarf-P-radiation
dwarf-P-cloudMicrophysics

Compute physics tendencies $\mathcal{P}(\tilde{\boldsymbol{u}}^{n+1})$

Compute coupling tendencies $\mathcal{C}(\tilde{\boldsymbol{u}}^{n+1})$

dwarf-D-advection

Grid-point
dwarf-D-GPderivative
dwarf-D-ellipticSolver

Spectral
dwarf-D-spectralTransform

Compute new predictor for future model state

$$\tilde{\tilde{\boldsymbol{u}}}^{n+1} = \boldsymbol{u}^n_d + \overline{\mathcal{P}}(\tilde{\boldsymbol{u}}^{n+1}) + \overline{\mathcal{C}}(\tilde{\boldsymbol{u}}^{n+1})$$

Solve elliptic Helmholtz problem and back-substitute to obtain model state at new time-level

$$\boldsymbol{u}^{n+1} = \tilde{\tilde{\boldsymbol{u}}}^{n+1} + \mathcal{H}elm(\tilde{\tilde{\boldsymbol{u}}}^{n+1})$$

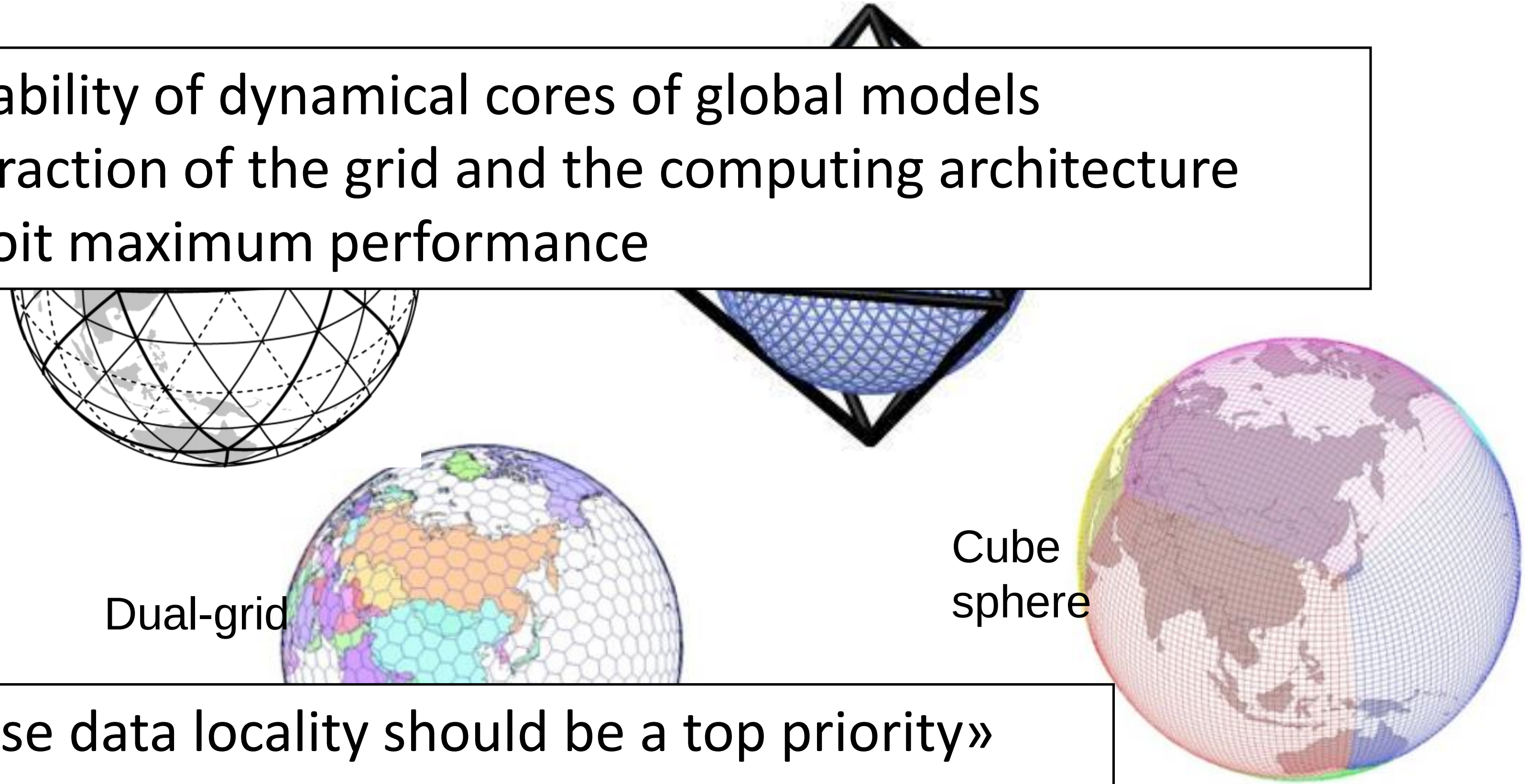
Peter Bauer

ECMWF · MeteoSwiss · DMI · RMI · NVIDIA · METEO FRANCE Toujours un temps d'avance · ICHEC · Deutscher Wetterdienst Wetter und Klima aus einer Hand · Optalysys · Bull atos technologies · PSNC · Loughborough University

UP - Preparing for future models

GridTools On Irregular Grids

- Portability of dynamical cores of global models
- Abstraction of the grid and the computing architecture
- Exploit maximum performance

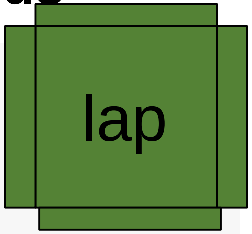

Dual-grid

Cube sphere

> «Increase data locality should be a top priority»
>
> Thomas Schulthess, HPC in Meteorology, 2016

Data Locality on unstructured meshes

Loop fusion and redundant computations on Lat-Lon grids



```
!OMP PARALLEL DO
DO iblock=1,nblocks
  DO k=1,nz
    DO j=jstart-1,jend+1
      DO i=istart-1,iend+1
        lap(i,j,k,iblock) =-4*u(i,j,k,iblock) + u(i+1,j,k,iblock) +
          u(i-1,j,k,iblock) + u(i,j+1,k,iblock) + u(i,j-1,k,iblock)
      ENDDO
    ENDDO
    DO j=jstart,jend
      DO i=istart,iend
        udiff(i,j,k,iblock) =-4*lap(i,j,k,iblock) + lap(i+1,j,k,iblock) +
          lap(i-1,j,k,iblock) + lap(i,j+1,k,iblock) + lap(i,j-1,k,iblock)
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

Funded by the European Union

Data Locality on unstructured meshes

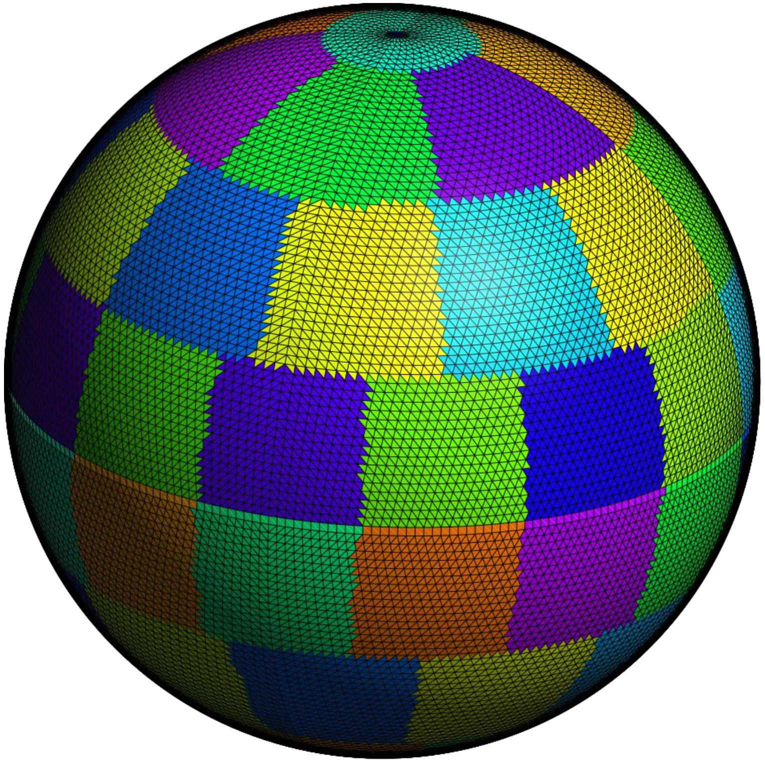
On irregular grids?

```
DO jb = i_startblk, i_endblk
   DO jc = i_startidx, i_endidx
      DO jk = slev, elev
          div(jc,jk,jb) = u(iid(ic,jb,1),jk,iblk(ic,jb,1)) * ptr_int%geofac_div(jc,1,jb) + &
                u(iid(ic,jb,2),jk,iblk(ic,jb,2)) * ptr_int%geofac_div(jc,2,jb) + &
                u(iid(ic,jb,3),jk,iblk(ic,jb,3)) * ptr_int%geofac_div(jc,3,jb)
      ENDDO
   ENDDO
ENDDO
Halo_exchange
DO jb = i_startbl
   DO jc = i_startidx, i_endidx
      DO jk = slev, elev
          div2 (jc,jk,jb) = div(iid(ic,jb,1),jk,iblk(ic,jb,1)) * ptr_int%geofac_div(jc,1,jb) + &
                div(iid(ic,jb,2),jk,iblk(ic,jb,2)) * ptr_int%geofac_div(jc,2,jb) + &
                div(iid(ic,jb,3),jk,iblk(ic,jb,3)) * ptr_int%geofac_div(jc,3,jb)
      ENDDO
   ENDDO
ENDDO
```

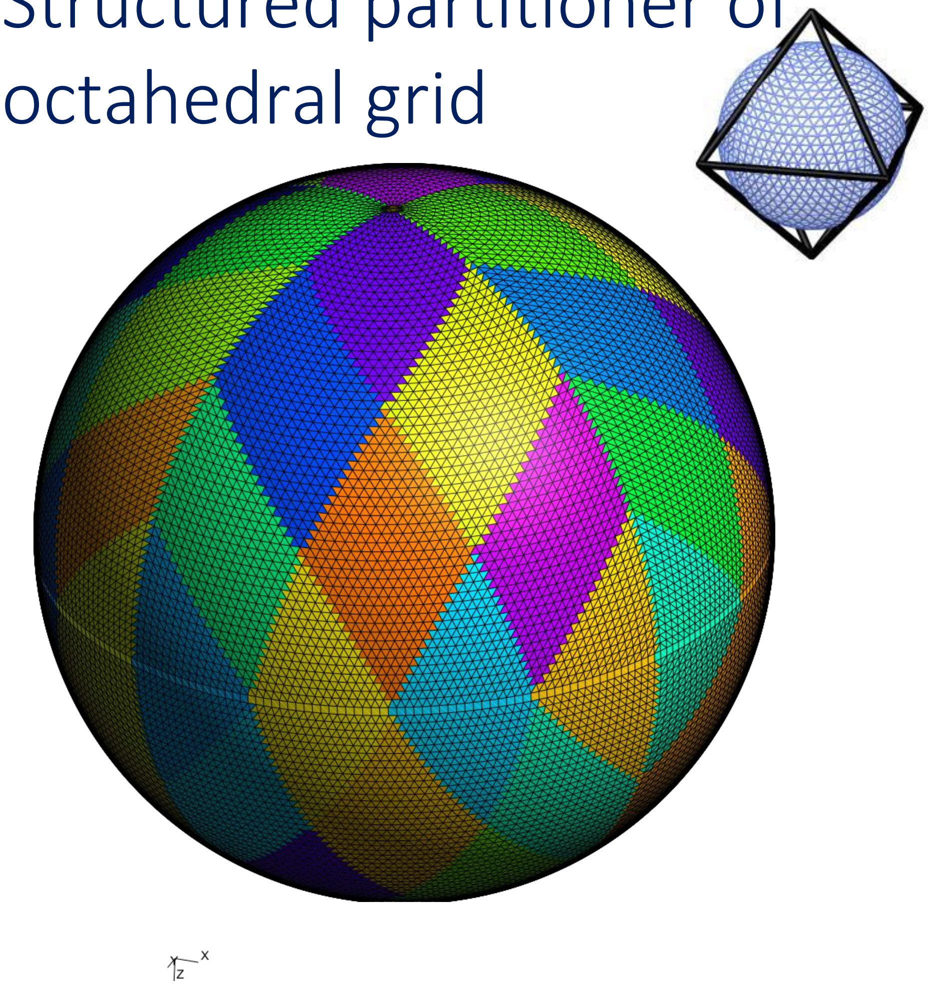
Are we abandoning Loop Fusion on unstructured meshes?

Funded by the European Union

Equal partitioner of octahedral grid

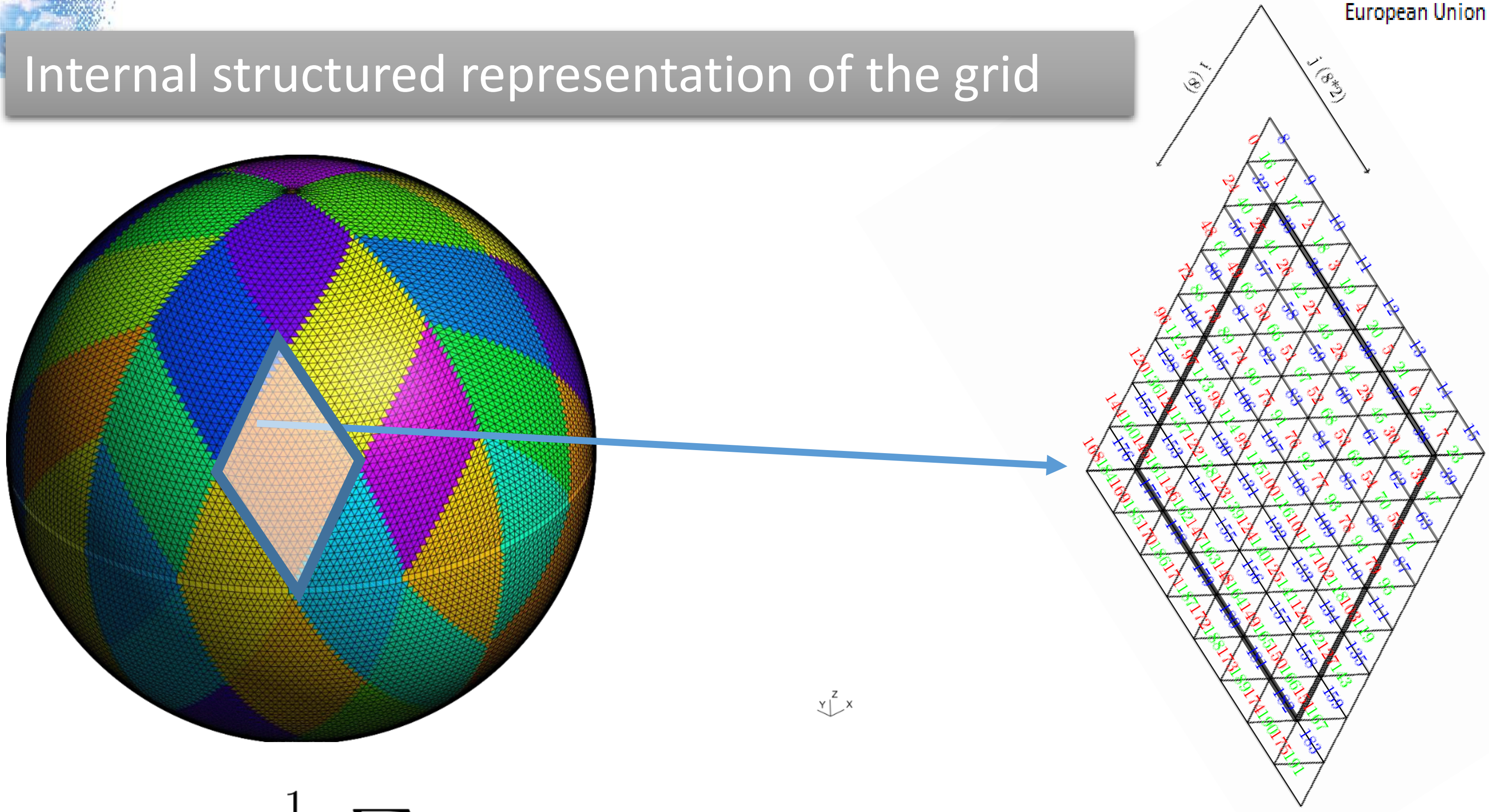


Structured partitioner of octahedral grid



Funded by the
European Union

Internal structured representation of the grid



$$\mathrm{div}(\mathbf{v})_i := \frac{1}{A_i} \sum_{l \in \mathcal{E}(i)} v_{n_l} (\mathbf{N}_l \cdot \mathbf{n}_{i,l}) l$$

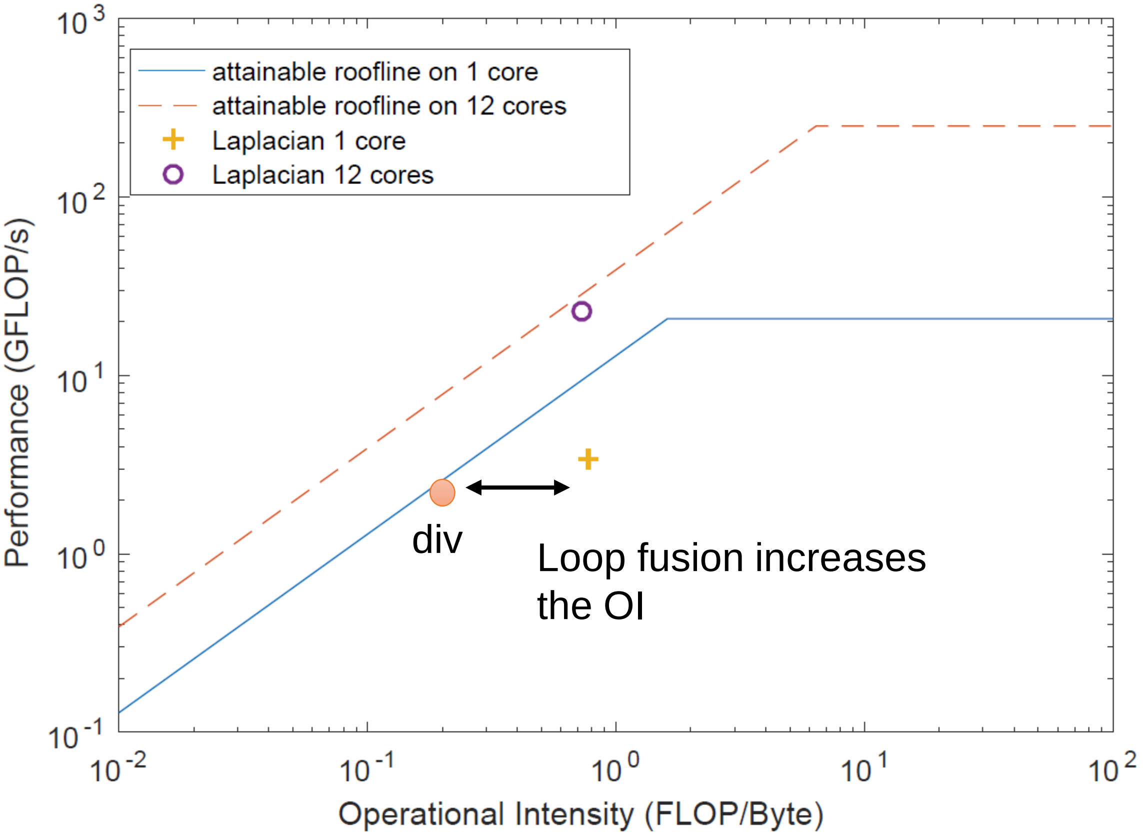
Unstructured DSL syntax

```
on_edges(sum_reduction, v(), l()) / A();
```

ECMWF
MeteoSwiss
DMI
RMI
NVIDIA
METEO FRANCE Toujours un temps d'avance
ICHEC Irish Centre for High-End Computing
Deutscher Wetterdienst Wetter und Klima aus einer Hand DWD
Optalysys
Bull atos technologies
PSNC
Loughborough University

Funded by the
European Union

$$(\nabla_d^2 \mathbf{v})_l \cdot \mathbf{N}_l := \mathrm{grad}_n \Big[\mathrm{div}(\mathbf{v})\Big]_l - \mathrm{grad}_\tau \Big[\mathrm{curl}(\mathbf{v})\Big]_l$$

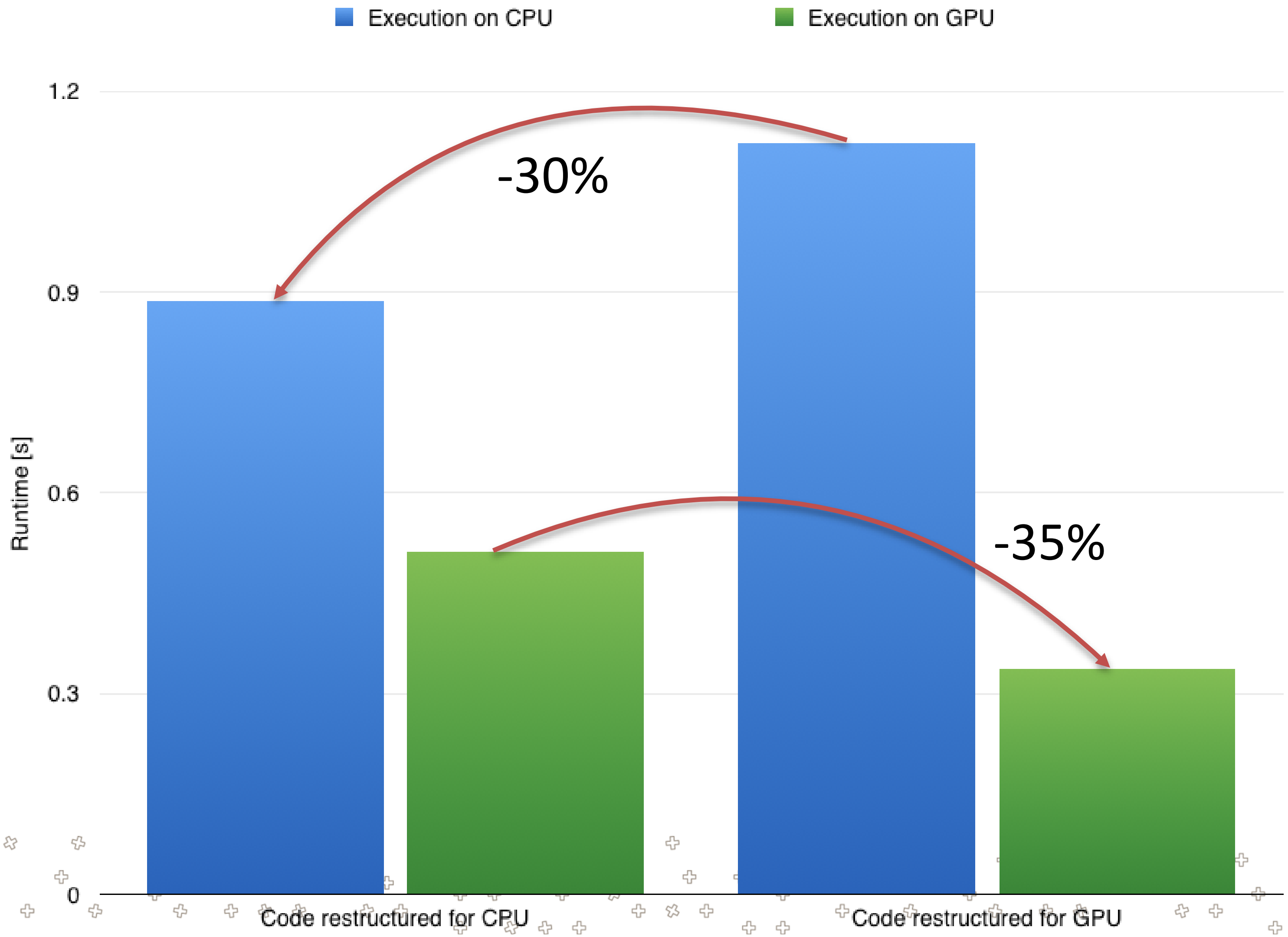


Loop fusion increases the OI

Domain size	1 core			12 cores			GPU	
	Fortran	GridTools	speedup	Fortran	GridTools	speedup	Gridtools	speedup
256*256*50	0.784	0.3829	**2.05x**	0.1279	0.0569	**2.25x**	0.0473	**2.7x**

DOWN - Preparing for future models

Performance portability: COSMO Radiation



CLAW: Low-level transformations

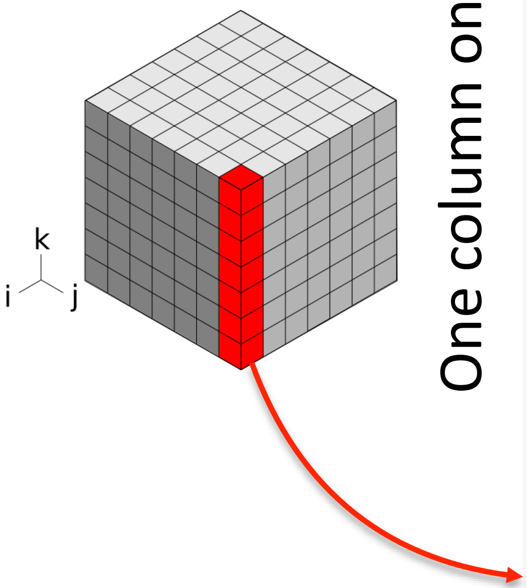
```
!$acc parallel loop
!$claw loop-interchange
DO k=1,nz
  !$claw loop-extract fusion
  CALL fct()
  !$claw loop-fusion group(j)
  !$acc loop
  DO j=1,nproma
    ! 1st loop body
  END DO
  !$claw loop-fusion group(j)
  !$acc loop
  DO j=1,nproma
    ! 2nd loop body
  END DO
  !$claw loop-fusion group(j)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

```
!$acc parallel loop
DO j=1,nproma
  !$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 One column abstraction

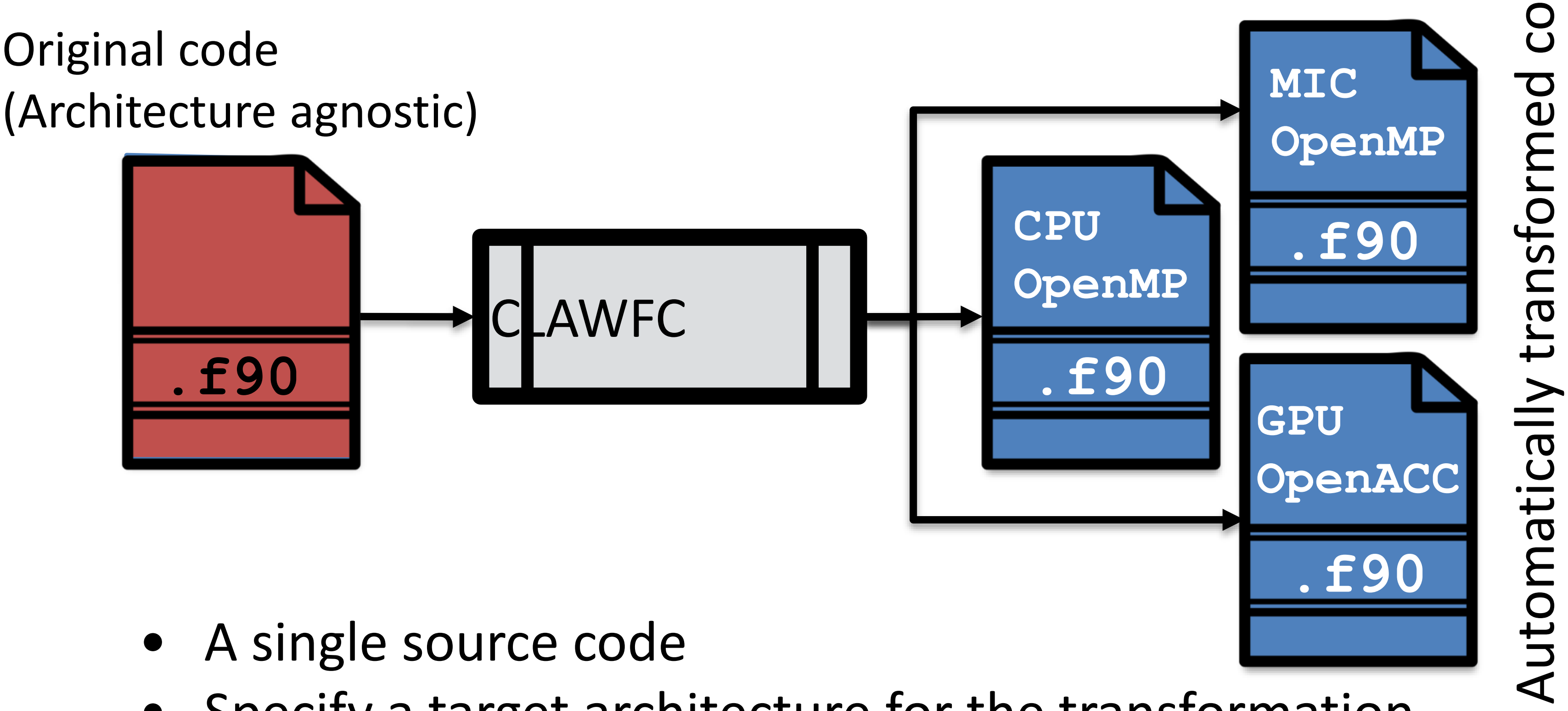


One column only

```
SUBROUTINE lw_solver(…)
     !$claw define dimension icol(1:ncol) &
     !$claw parallelize
     DO igpt = 1, ngpt
       DO ilev = 1, nlay
              tau_loc(ilev) = max(tau(ilev,igpt) …
         END DO
         DO ilev = 1, nlay
              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
```

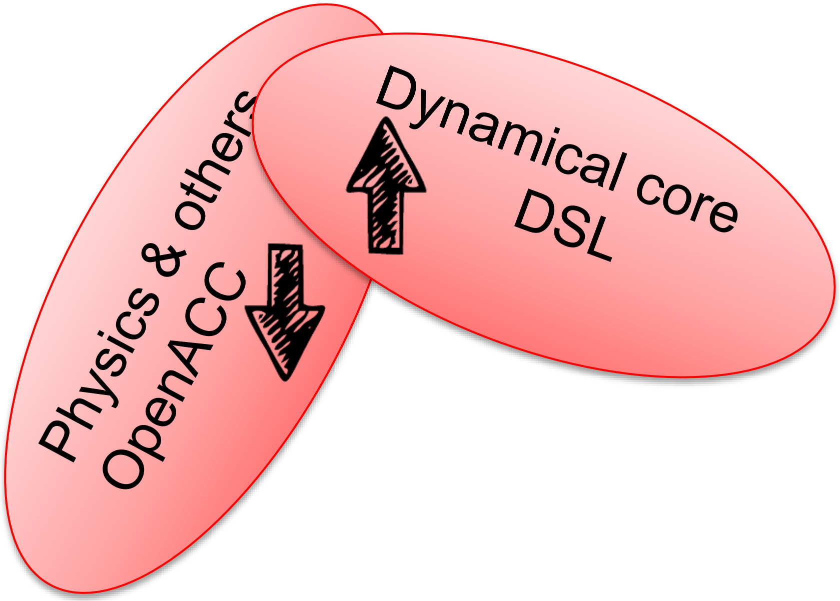
```
clawfc --directive=openmp --target=cpu -o mo_lw_solver.omp.f90 mo_lw_solver.f90
```

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

pictures: Por Suppasit from Noun Project

Conclusions

- New setup COSMO-1 and COSMO-E running on hybrid GPU system
- About 4x was gained by moving to GPUs as compare to traditional CPUs



Performance comparison with CPU only system

Piz Dora (Cray XC40)

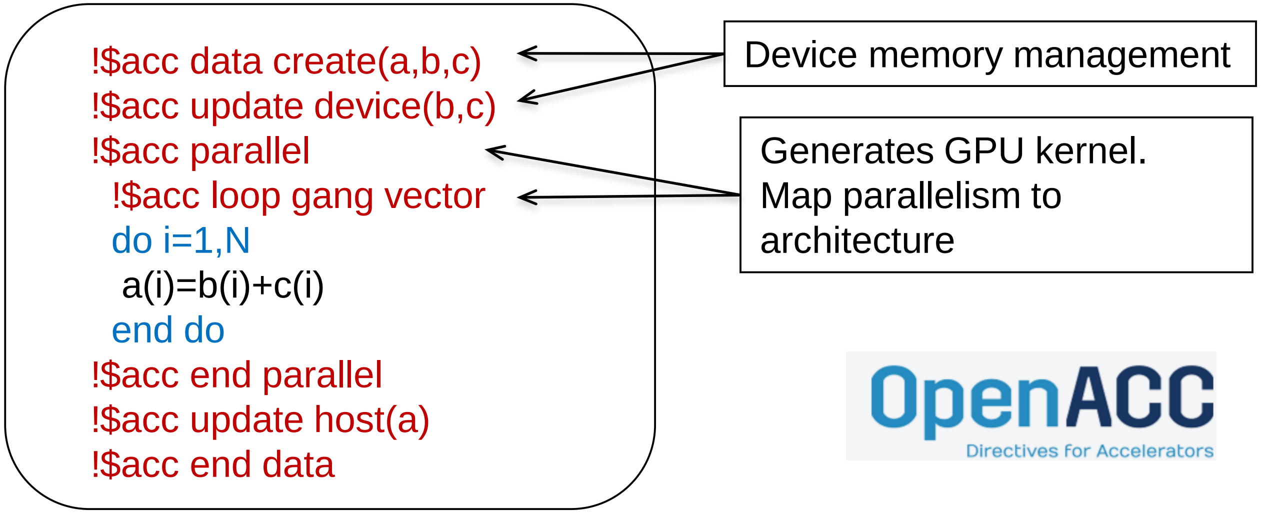
- “Traditional” CPU based system
- Compute nodes with 2 Intel Xeon E5-2690 v3 socket (Haswell)
- Pure compute rack
- Rack has 192 compute nodes
- Very high density (supercomputing line)

- Performance comparison for the COSMO-E ensemble members

Results

	Piz Dora	**Piz Kesch**	Factor
Energy per member	10 kWh	2.1 kWh	**4.8 x**
Time with 8 sockets per member	3.9 h	1.0 h	**3.9 x**
Cabinets required to run ensemble at required time-to-solution	1.4	0.38	**3.8 x**

Porting with OpenACC directives



```
!$acc data create(a,b,c)
!$acc update device(b,c)
!$acc parallel
  !$acc loop gang vector
  do i=1,N
   a(i)=b(i)+c(i)
  end do
!$acc end parallel
!$acc update host(a)
!$acc end data
```

Device memory management

Generates GPU kernel.
Map parallelism to
architecture

- Directive-based programming model for C++ and Fortran
- Provides abstraction to define parallel region, data locality (GPU, CPU) and mapping to specific hardware parallelism (gang, worker, vector)
- Enable to port large codes to GPU with acceptable efforts
- Used for porting the physics, assimilation, I/O.

Performance portability with OpenACC

- In some cases CPU and GPU have different optimization requirement

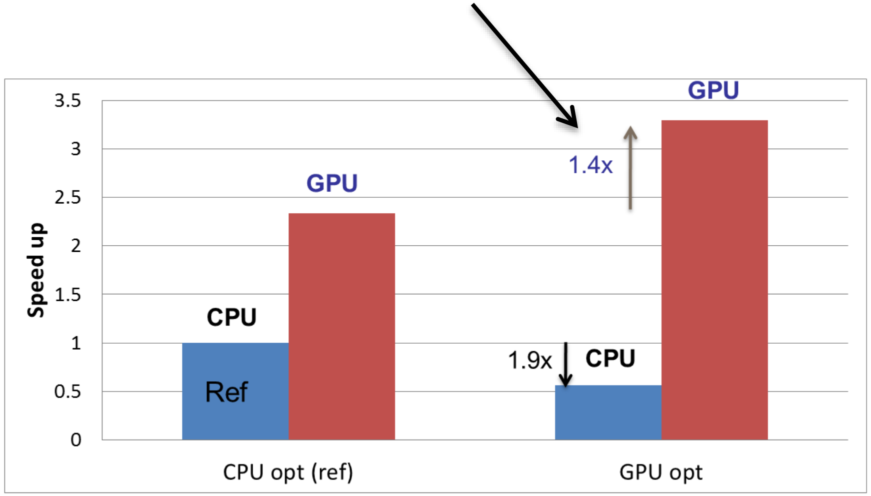
CPU : compute bound

- Auto-vectorization : small loop
- Pre-computation

GPU : memory bound limited

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

Optimize code for GPU runs 1.9x slower on CPU



Radiation parametrization, domain 128x128x60, 1 Sandy Bridge CPU vs 1 K20x CPU

- Current solution : different code paths for time critical components with pre-processor macros

CLAW approach

- Goal : Provide language abstraction for performance portability in climate and weather model
- Directives with code transformation

```
SUBROUTINE inv_th(pclc,pca1, …)
  INTEGER:: ki1sd

  !$acc parallel
  !$acc loop collapse(3)
  !$claw loop-interchange (k,i,j)
  DO i=istart,iend
    DO j=jstart,jend
      DO k=kstart,kend
        ! Computation is done here
      END DO
    END DO
  END DO
  !$acc end parallel

END SUBROUTINE inv_th
```

CLAW

- Code manipulation with AST
- Based on the OMNI compiler
- Transformed code can be compile with standard compiler

CLAW language definition are available on github :

https://github.com/C2SM-RCM/claw-language-definition

References

Lapillonne, X. and Fuhrer, O. (2014). Using compiler directives to port large scientific applications to GPUs: An example from atmospheric science. *Parallel Processing Letters*, *24*(01), 1450003.

Fuhrer, O. and *al.* (2014). Towards a performance portable, architecture agnostic implementation strategy for weather and climate models. *Supercomputing frontiers and innovations*, *1*(1), 45-62.

Gysi, T. and *al.* (2015). STELLA: a domain-specific tool for structured grid methods in weather and climate models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (p. 41). ACM.

Thank you

Increase of x40 in computational cost of operational setup

Key ingredients

Processor performance (Moore’s law)	~2.8 x
Code refactoring and port to GPUs	~3.9 x
Increase utilization of system	~2.8 x
Increase in number of sockets	~1.3 x

- Target system architecture to application

Image: Cray