ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

GPU Consideration for Next Generation Weather (and Climate) Simulations

Oliver Fuhrer[1], Tobias Gisy[2], Xavier Lapillonne[3], Will Sawyer[4], Ugo Varetto[4], Mauro Bianco[4], David Müller[2], and Thomas C. Schulthess[4,5,6]

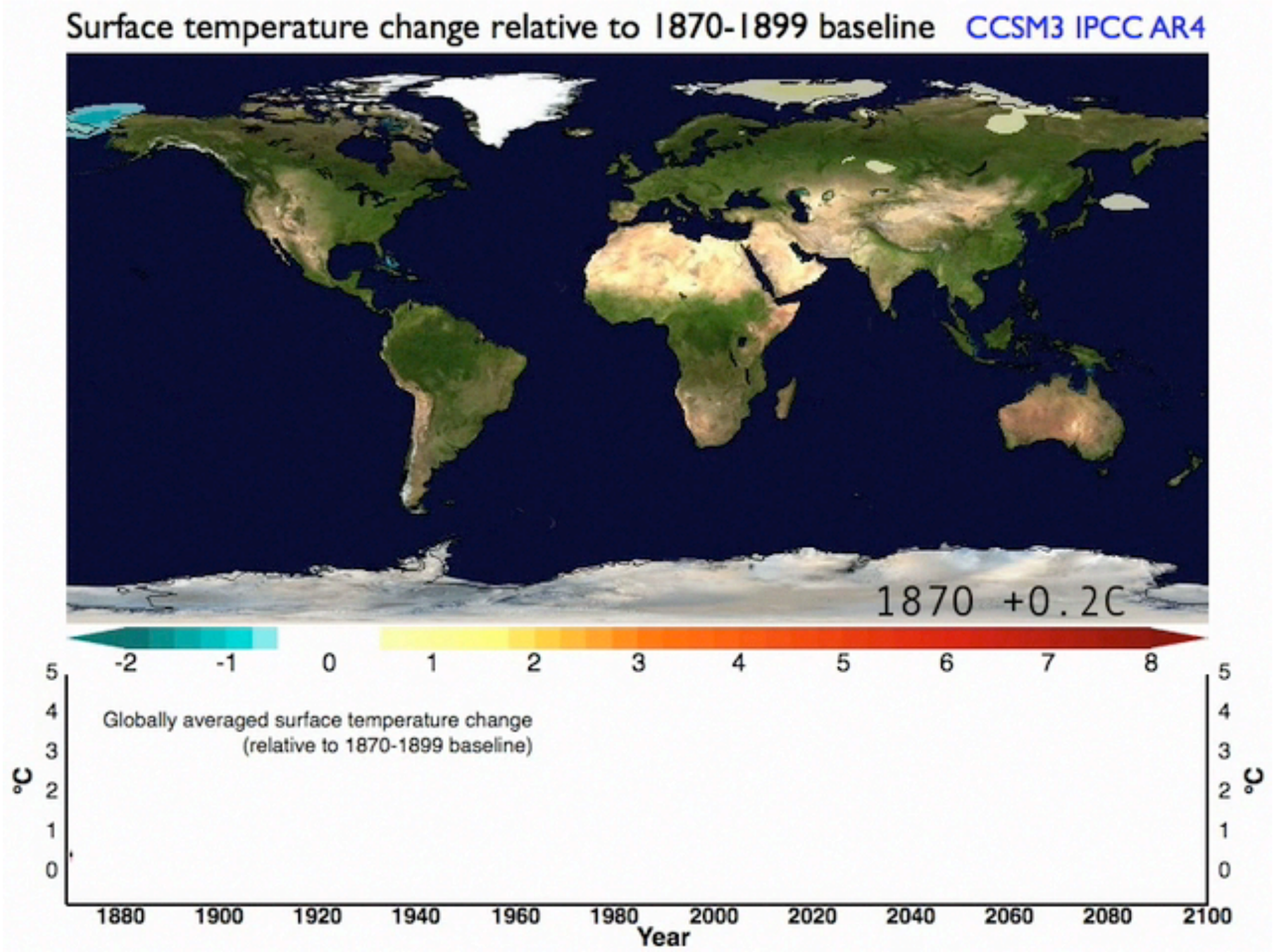
[1]Swiss Federal Office of Meteorology and Climatology; [2]Supercomputing Systems;
[3]Center for Climate Systems Modeling, ETH Zurich; [4]Swiss National Supercomputing Center
[5]Institute for Theoretical Physics, ETH Zurich; [6]Computer Science and Mathematics Division, ORNL

PLEASE NOTE THAT ...

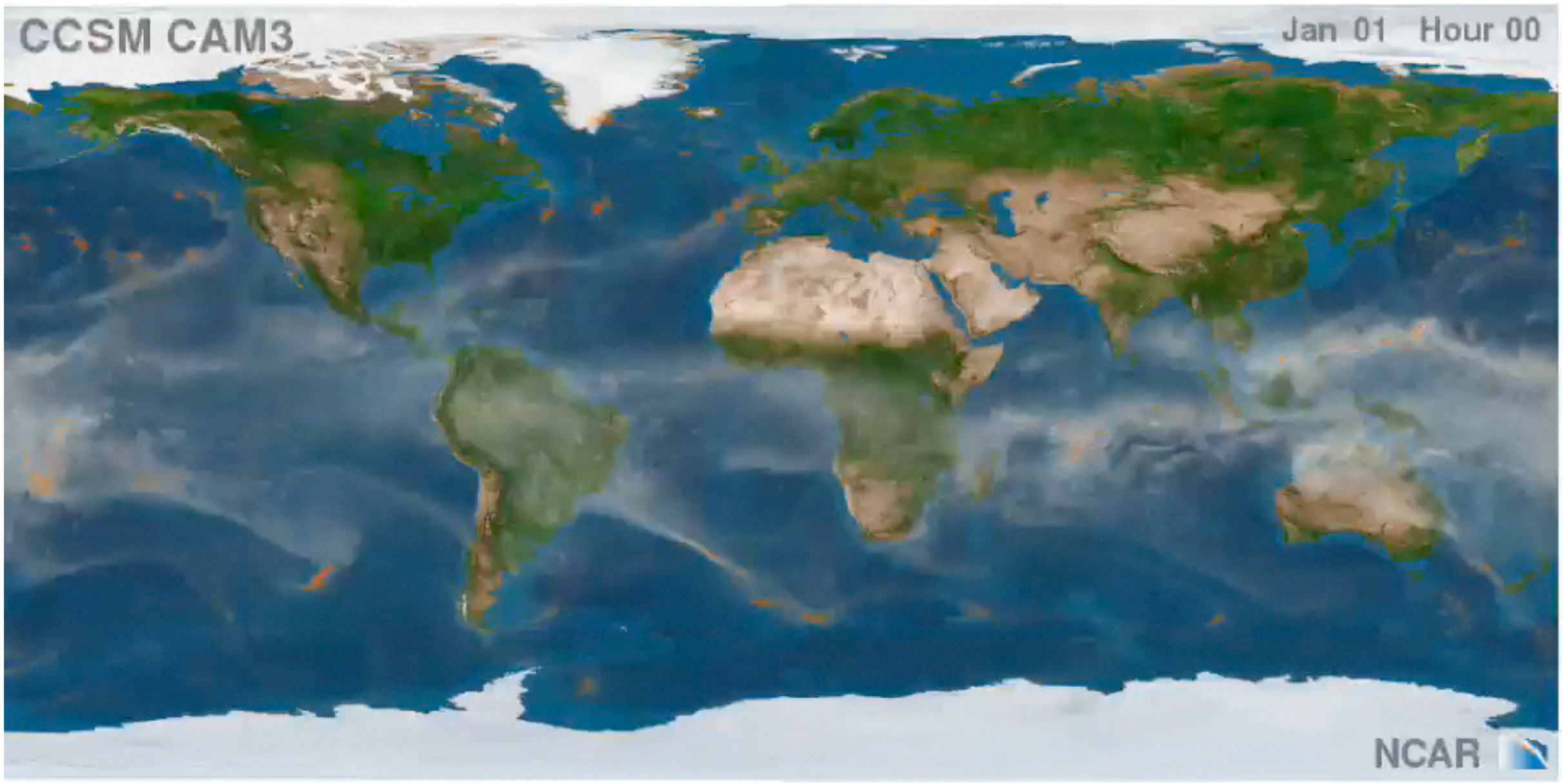
1. I am a computational physicist and have no research background in atmospheric science – please do not expect to receive deep insights in climate science or meteorology during my presentation

2. I report on work in progress of a project that runs through 12/2012 – many results I present are preliminary

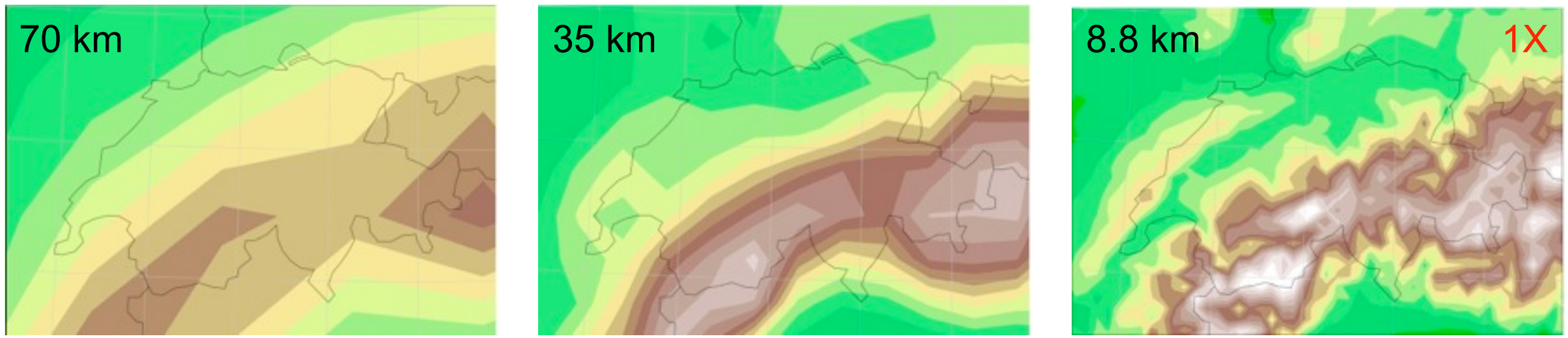
Today's state of the art climate simulation (resolution T85 ~ 148 km)



Experimental climate running at higher resolution (resolution T341 ~ 37 km)



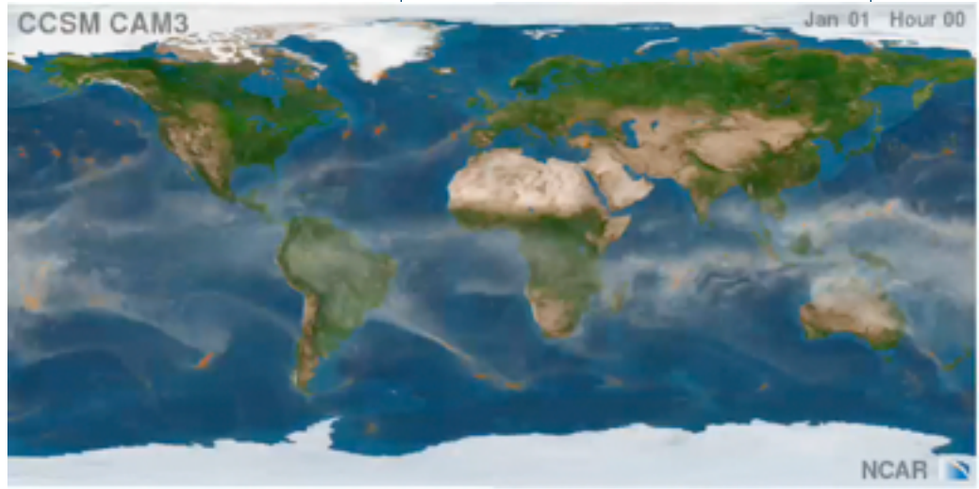
Why resolution is such an issue for Switzerland



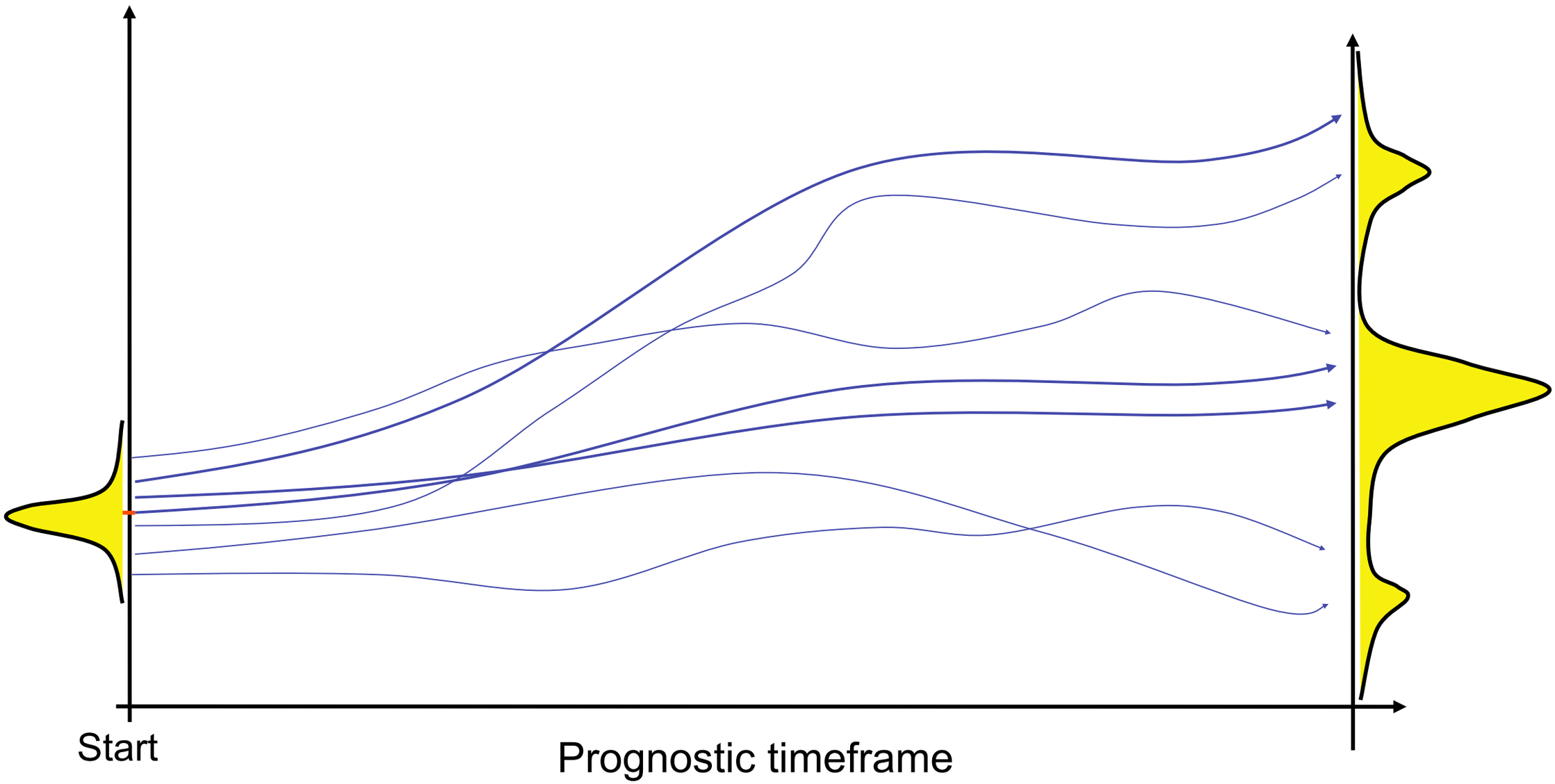
Source: Oliver Fuhrer, MeteoSwiss

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prognostic uncertainty



The weather system is chaotic
→ rapid growth of small perturbations (butterfly effect)



Ensemble method: compute distribution over many simulations

Source: Oliver Fuhrer, MeteoSwiss

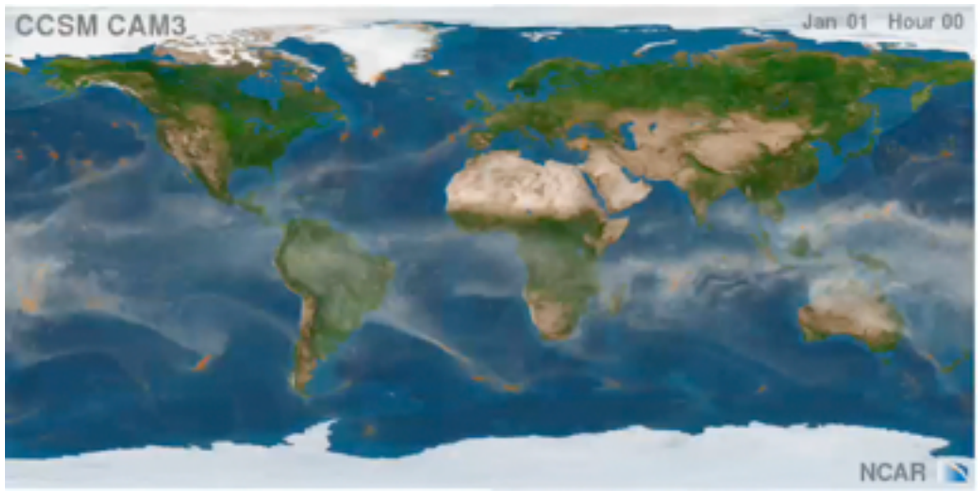
WHAT NEEDS TO BE IMPROVED?

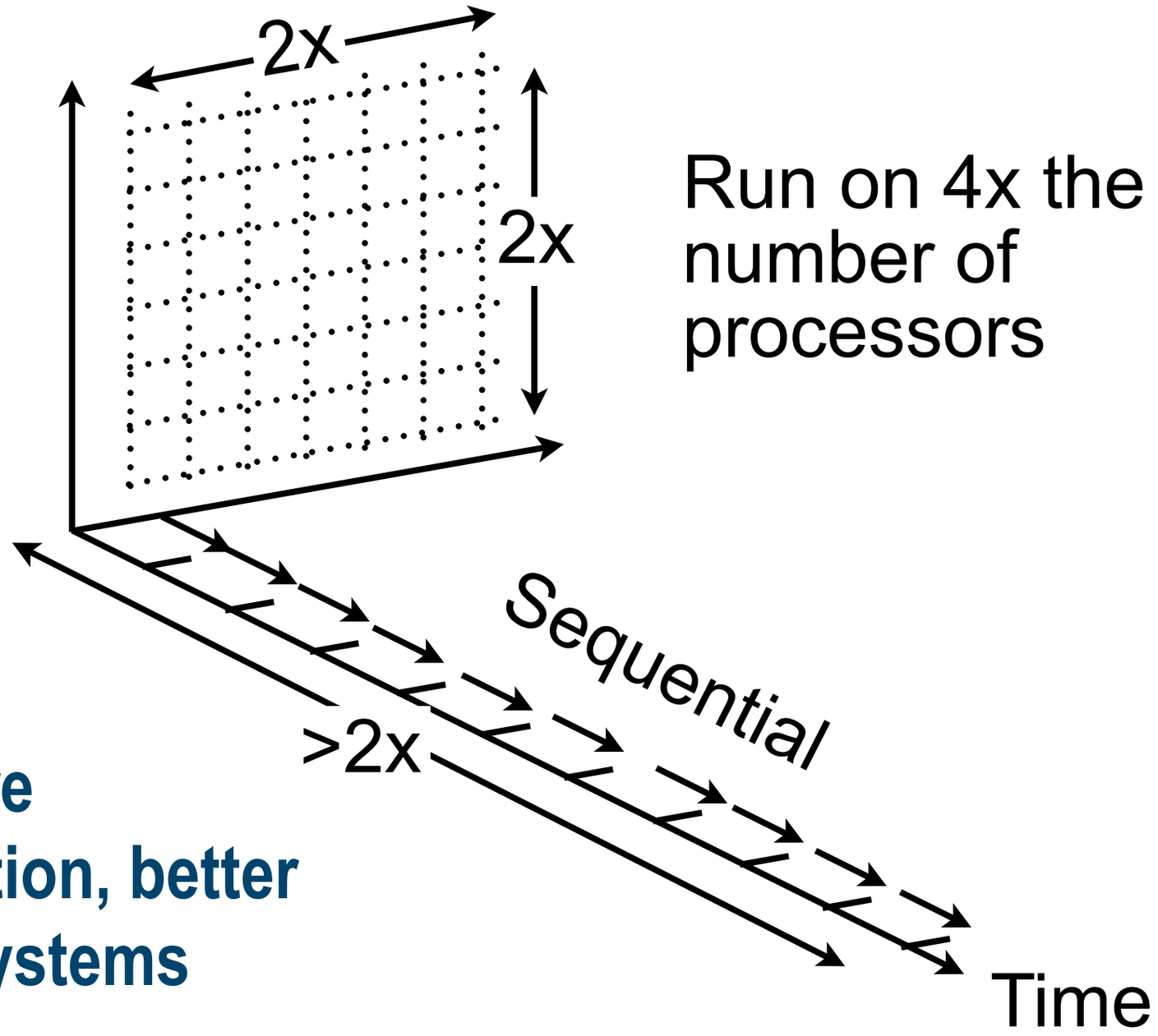
Higher resolution

Larger ensemble

Can we have both?

Increasing resolution (weak scaling): larger parallel computers only solve part of the problem





Calculations have to be more efficient: better implementation, better algorithms, more suitable systems

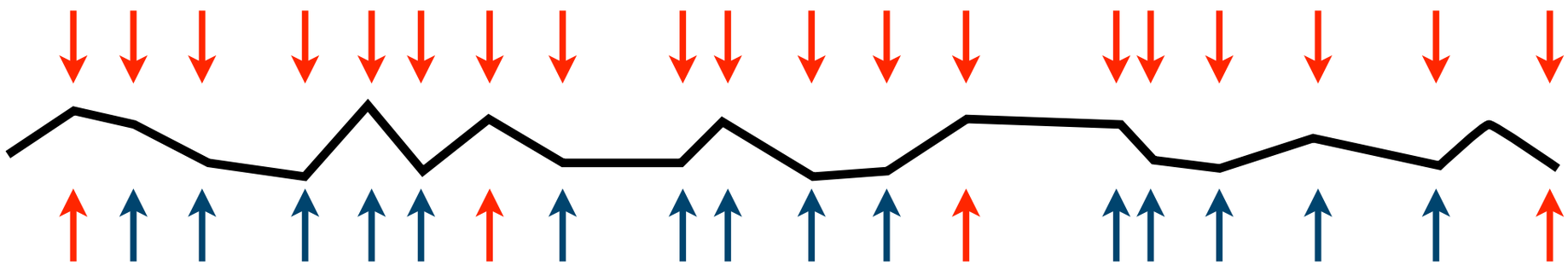
Applications running at scale on Jaguar @ ORNL (Spring 2011)

Domain area	Code name	Institution	# of cores	Performance	Notes
Materials	DCA++	ORNL	213,120	**1.9 PF**	2008 Gordon Bell Prize Winner
Materials	WL-LSMS	ORNL/ETH	223,232	**1.8 PF**	2009 Gordon Bell Prize Winner
Chemistry	NWChem	PNNL/ORNL	224,196	1.4 PF	2008 Gordon Bell Prize Finalist
Materials	DRC	ETH/UTK	186,624	1.3 PF	2010 Gordon Bell Prize Hon. Mention
Nanoscience	OMEN	Duke	222,720	> 1 PF	2010 Gordon Bell Prize Finalist
Biomedical	MoBo	GaTech	196,608	780 TF	2010 Gordon Bell Prize Winner
Chemistry	MADNESS	UT/ORNL	140,000	550 TF	
Materials	LS3DF	LBL	147,456	442 TF	2008 Gordon Bell Prize Winner
Seismology	SPECFEM3D	USA (multiple)	149,784	165 TF	2008 Gordon Bell Prize Finalist
Combustion	S3D	SNL	147,456	83 TF	
Weather	WRF	USA (multiple)	150,000	50 TF	

Hirsch-Fey quantum Monte Carlo with Delayed updates (or Ed updates)

Ed D'Azevedo, ORNL

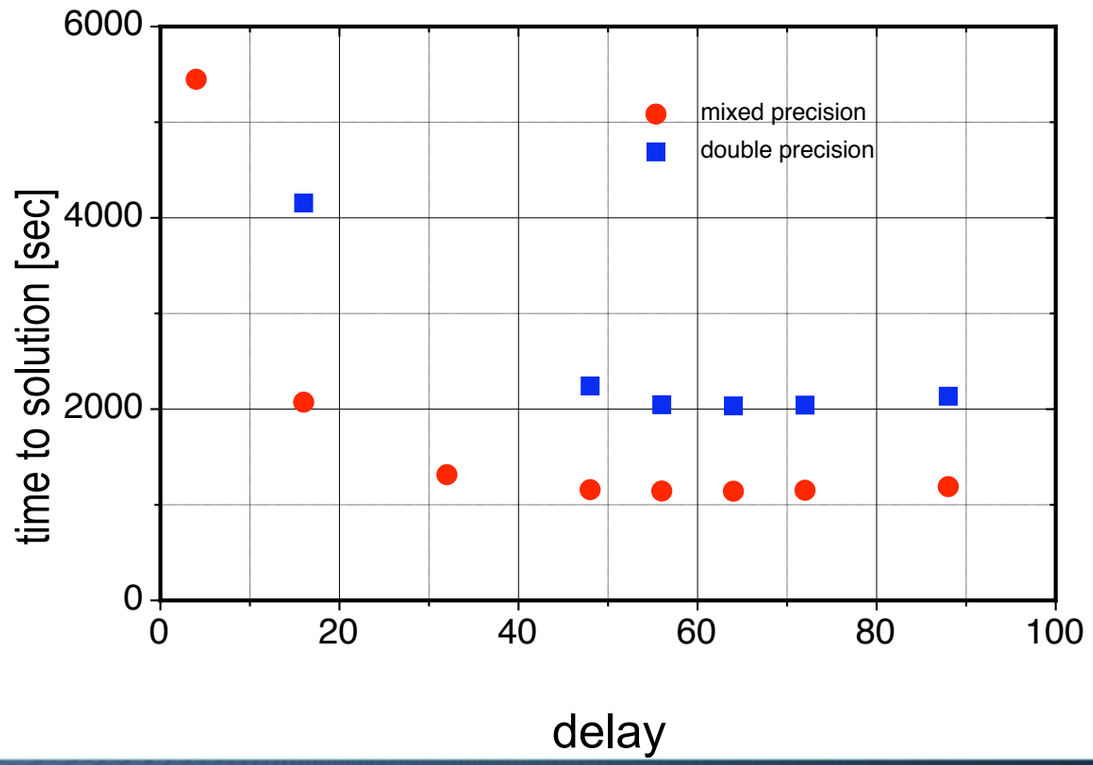
$$\mathbf{G}_c(\{s_i,l\}_{k+1}) = \mathbf{G}_c(\{s_i,l\}_k) + \mathbf{a}_k \times \mathbf{b}_k^t$$



$$\mathbf{G}_c(\{s_i,l\}_{k+1}) = \mathbf{G}_c(\{s_i,l\}_0) + [\mathbf{a}_0|\mathbf{a}_1|...|\mathbf{a}_k] \times [\mathbf{b}_0|\mathbf{b}_1|...|\mathbf{b}_k]^t$$

Complexity for *k* updates remains $\mathcal{O}(kN_t^2)$

But we can replace *k* rank-1 updates with one matrix-matrix multiply plus some additional bookkeeping.



Arithmetic intensity of a computation

$$\alpha = \frac{\text{floating point operations}}{\text{data transferred}}$$

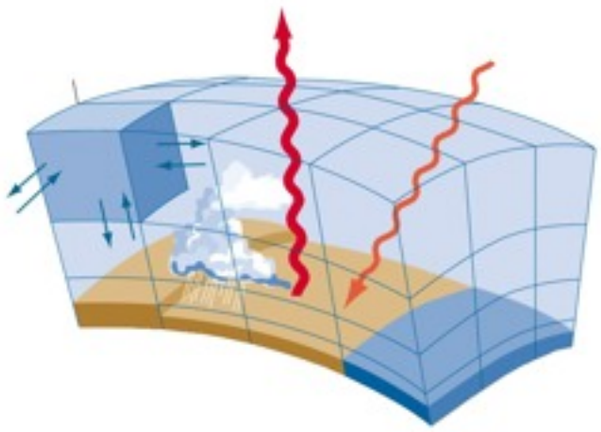
$$\mathbf{G}_c(\{s_i, l\}_{k+1}) = \mathbf{G}_c(\{s_i, l\}_k) + \mathbf{a}_k \times \mathbf{b}_k^t$$

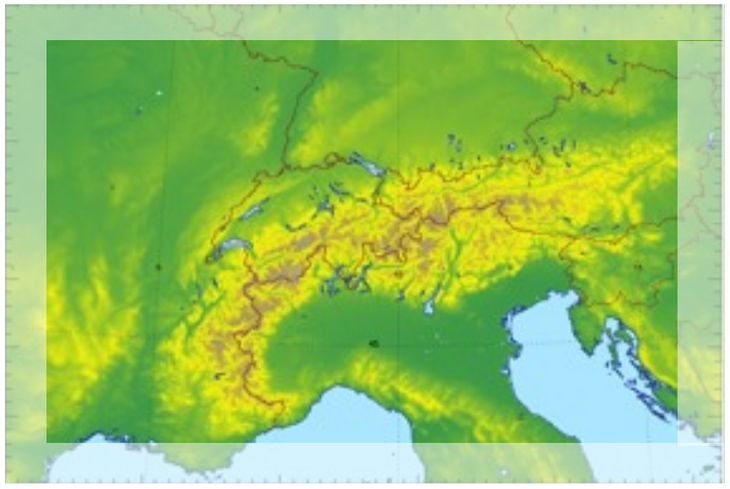
$$\alpha \approx o(1)$$

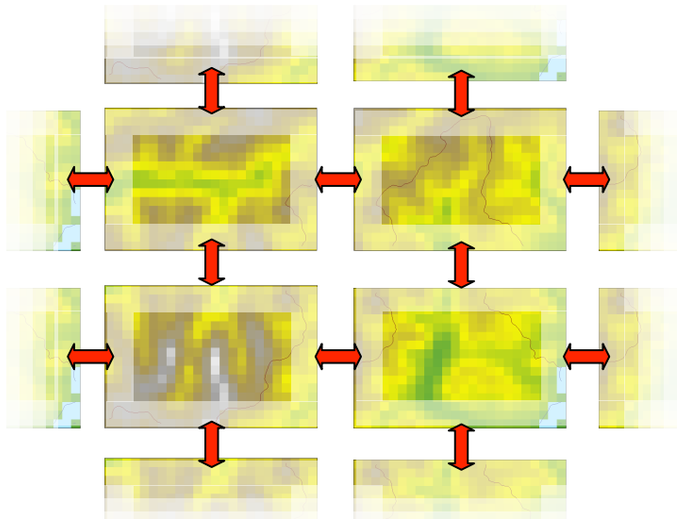
$$\mathbf{G}_c(\{s_i, l\}_{k+1}) = \mathbf{G}_c(\{s_i, l\}_0) + [\mathbf{a}_0|\mathbf{a}_1|...|\mathbf{a}_k] \times [\mathbf{b}_0|\mathbf{b}_1|...|\mathbf{b}_k]^t$$

$$\alpha \approx o(k)$$

Finite difference stencil to solve PDE on structured grid – climate, meteorology, seismic imaging, combustion

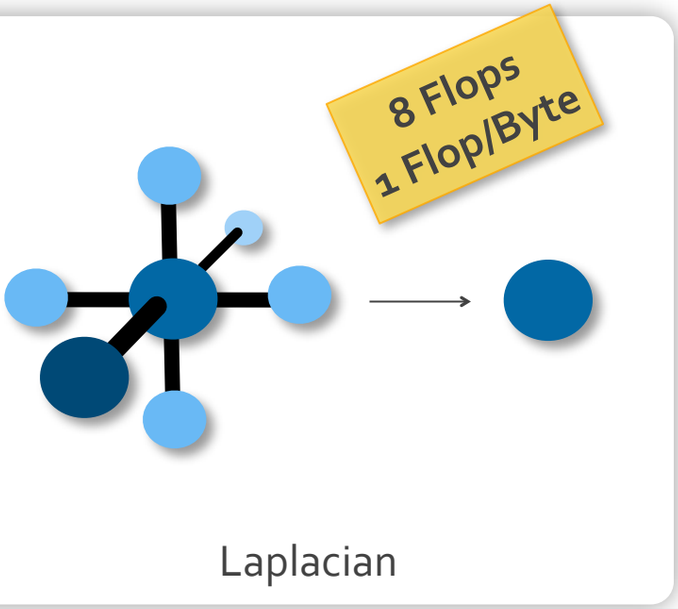


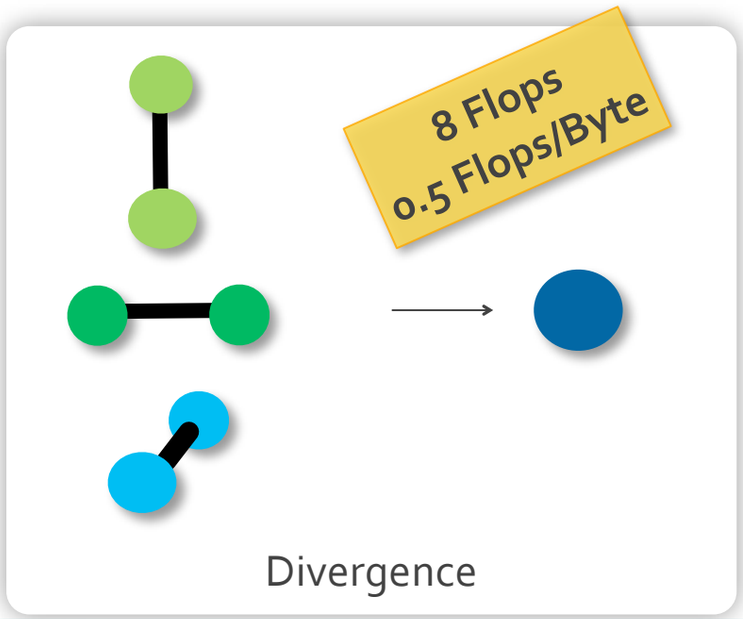


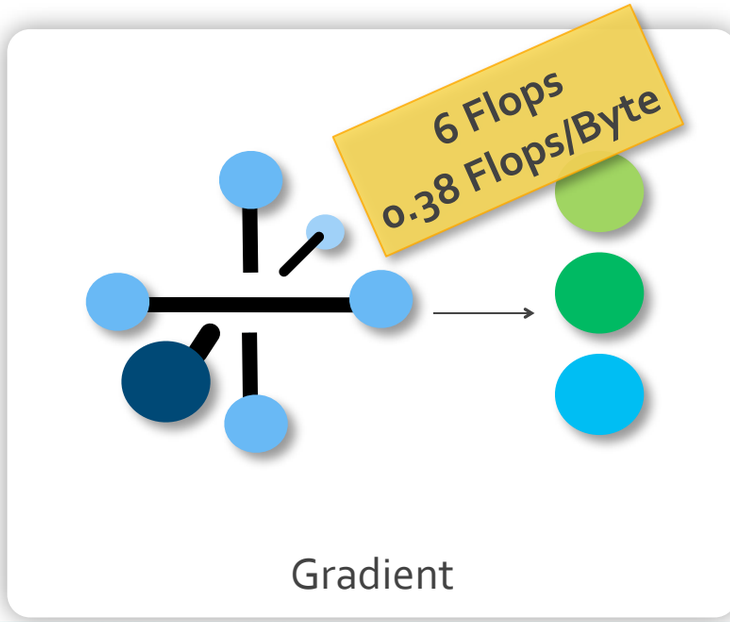


Partial differential equations on structured grid

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$







Algorithmic motifs and their arithmetic intensity

Arithmetic intensity: number of operations per word of memory transferred

Finite difference / stencil
in S3D and WRF (& COSMO)

Rank-1 update in HF-QMC → **Rank-N update in DCA++**

QMR in WL-LSMS

Sparse linear algebra
Matrix-Vector
Vector-Vector
BLAS1&2

Fast Fourier Transforms
FFTW & SPIRAL

Linpack (Top500)
Dense Matrix-Matrix
BLAS3

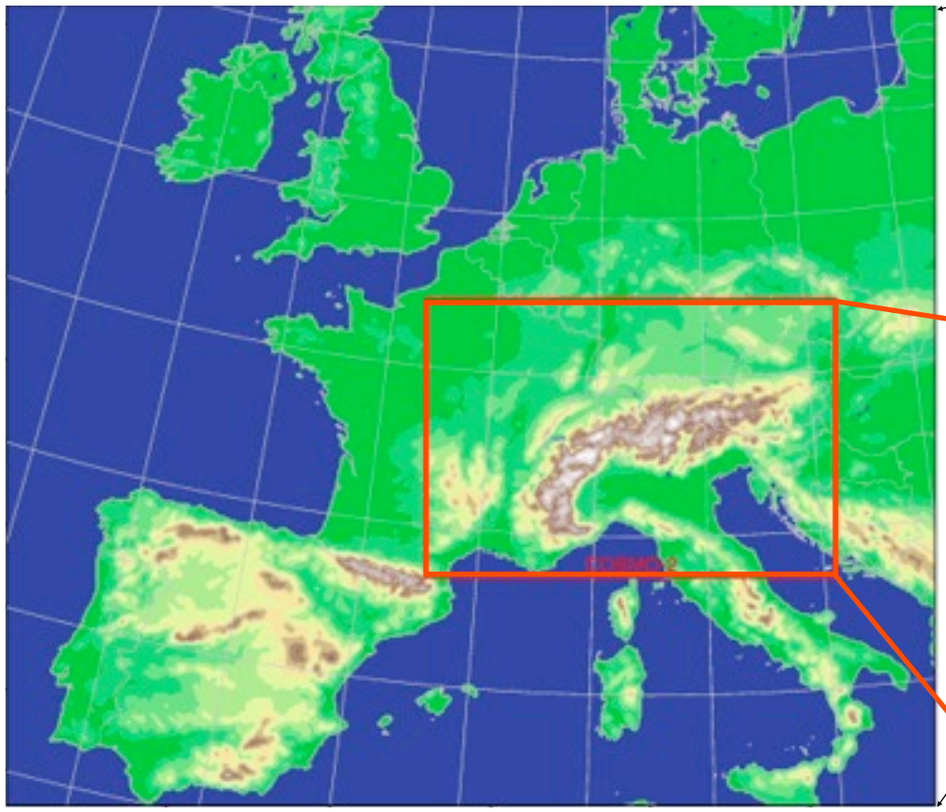
O(1) **O(log N)** **O(N)**

Supercomputers are designed for certain algorithmic motifs – which ones?

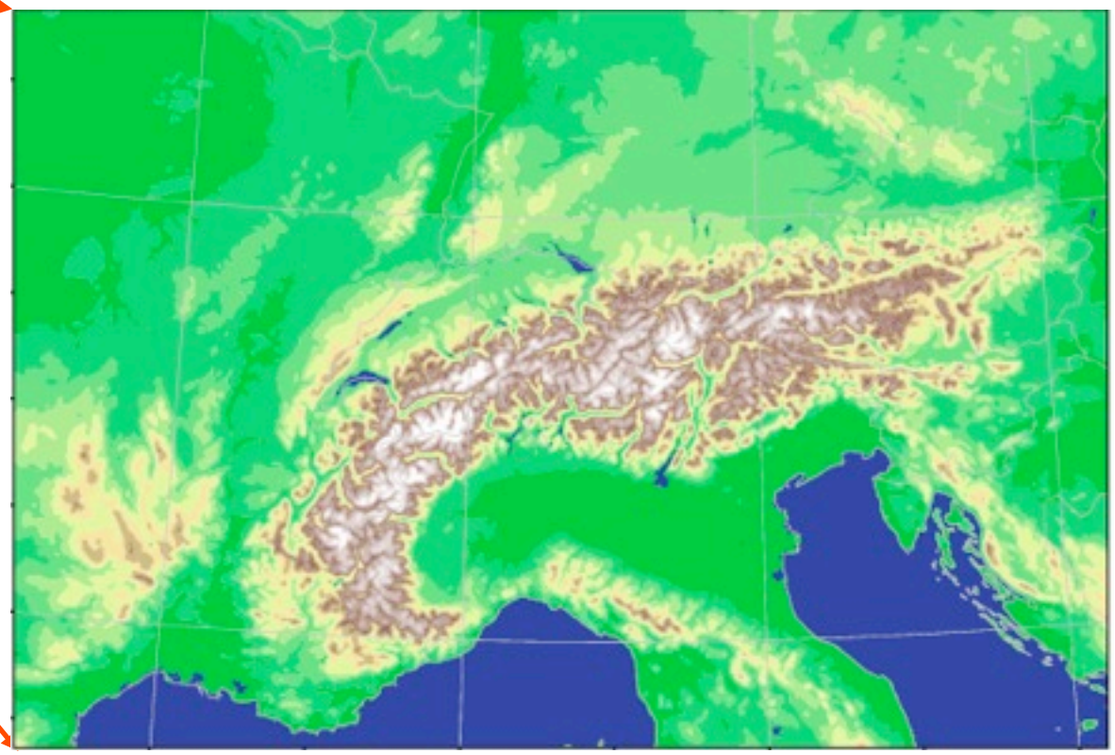
Numerical weather predictions for Switzerland (MeteoSwiss)

ECMWF: boundary conditions
16km, 91 layers
2x per day

COSMO-7 : 3x per day 72h forecast
6.6 km lateral grid, 60 layers



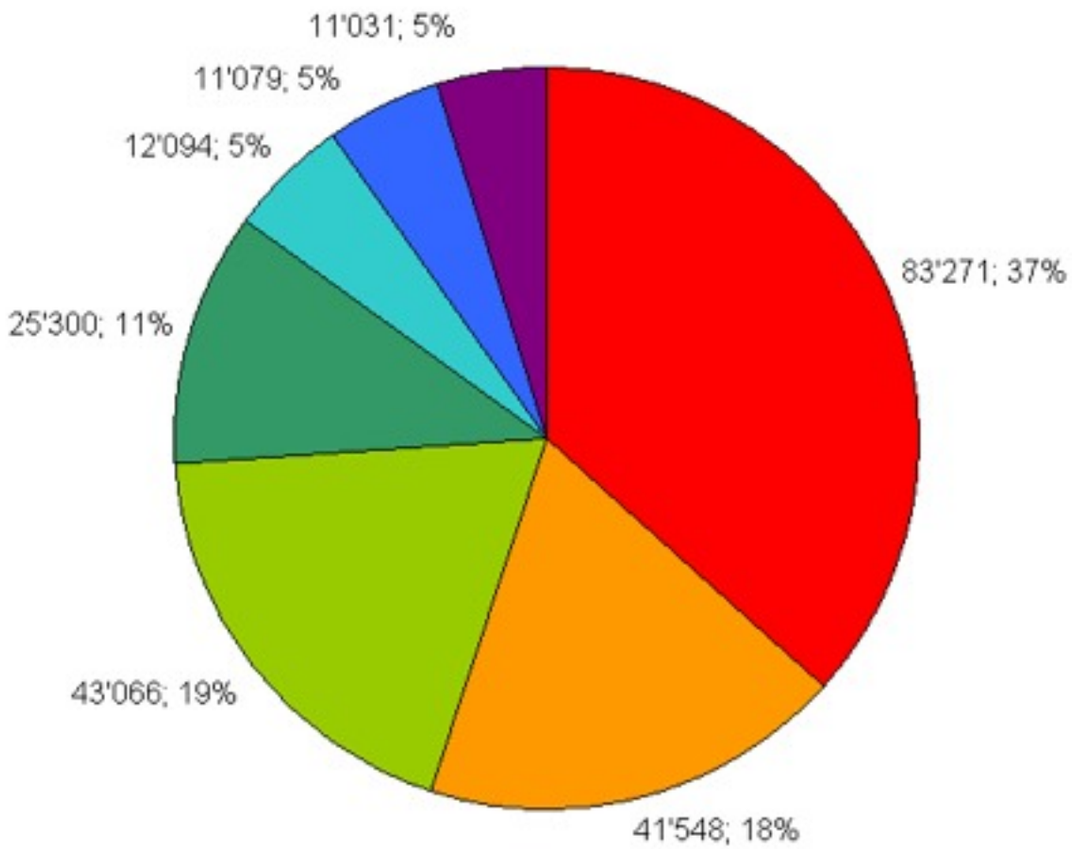
COSMO-2: 8x per day 24h forecast
2.2 km lateral grid , 60 layers



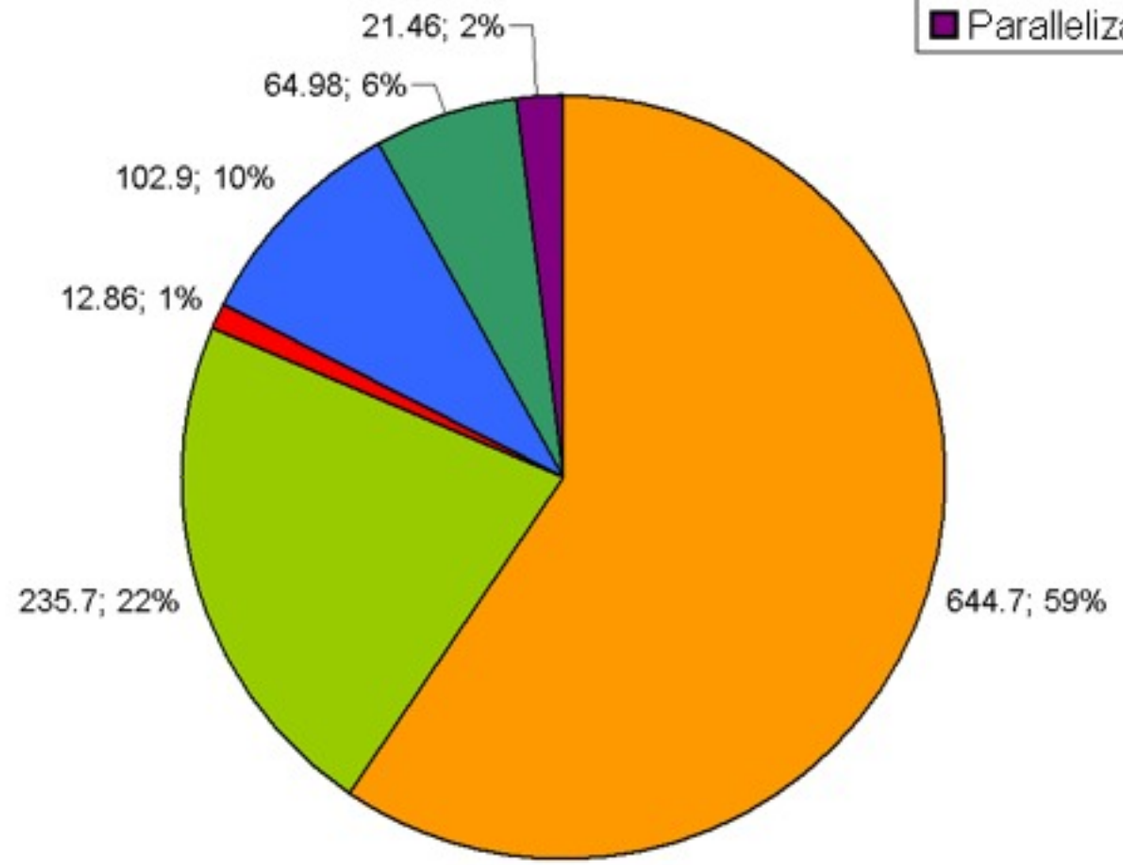
Performance profile of (original) COSMO-CCLM

Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90)



% Runtime



Source: Oliver Fuhrer, MeteoSwiss

Example of motif in the physics code

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i)**b(i) + sin(b(i)) * log(a(i))
  end do
end do
```

$$c = a^b + sin(b) \cdot log(a)$$

(niter=48; nwork=4096000)

Processor	Barcelona	Istambul	Magny-Cours	Interlagos
# cores	4 @ 2.6GHz	6 @ 2.4 GHz	12 @ 2.2 GHz	16 @ 2.1 GHz
Single core	17.8 s	17.3 s	18.8 s	19.9 s
All cores	4.5 s	2.9 s	1.61 s	1.31 s
Speedup	**3.96**	**5.97**	**11.7**	**15.19**
	1.55 (1.38)	1.80 (1.83)	1.23 (1.27)	

Example of motif in the dynamics code

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i) + b(i) * ( a(i+1) - 2.0d0*a(i) + a(i-1) )
  end do
end do
```

$$\frac{\partial a}{\partial t} = k\frac{\partial^2 a}{\partial x^2}$$

Processor	Barcelona	Istambul	Magny-Cours	Interlagos
# cores	4 @ 2.6GHz	6 @ 2.4 GHz	12 @ 2.2 GHz	16 @ 2.1 GHz
Single core	0.80 s	0.84 s	0.63 s	0.65 s
All cores	0.56 s	0.46 s	0.18 s	0.16 s
Speedup	**1.4**	**1.8**	**3.5**	**4.1**

1.22 (1.38) **2.56 (1.83)** **0.13 (1.27)**

Analyzing the two examples – how are they different?

Physics

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i)**b(i) + sin(b(i)) * log(a(i))
  end do
end do
```

3 memory accesses
136 FLOPs
➔ compute bound

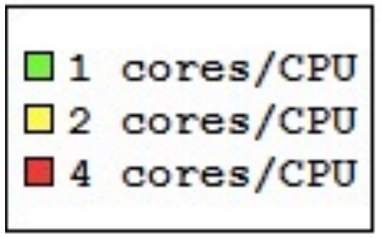
Dynamics

```
do j = 1, niter
  do i = 1, nwork
    c(i) = a(i) + b(i) * ( a(i+1) - 2.0d0*a(i) + a(i-1) )
  end do
end do
```

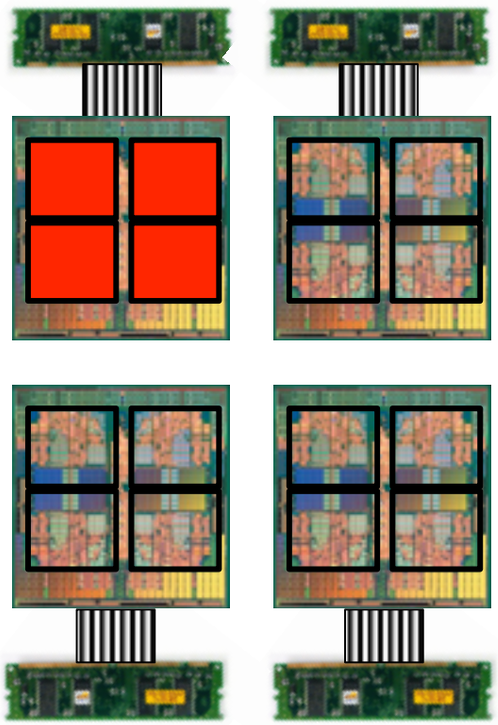
3 memory accesses
5 FLOPs
➔ memory bound

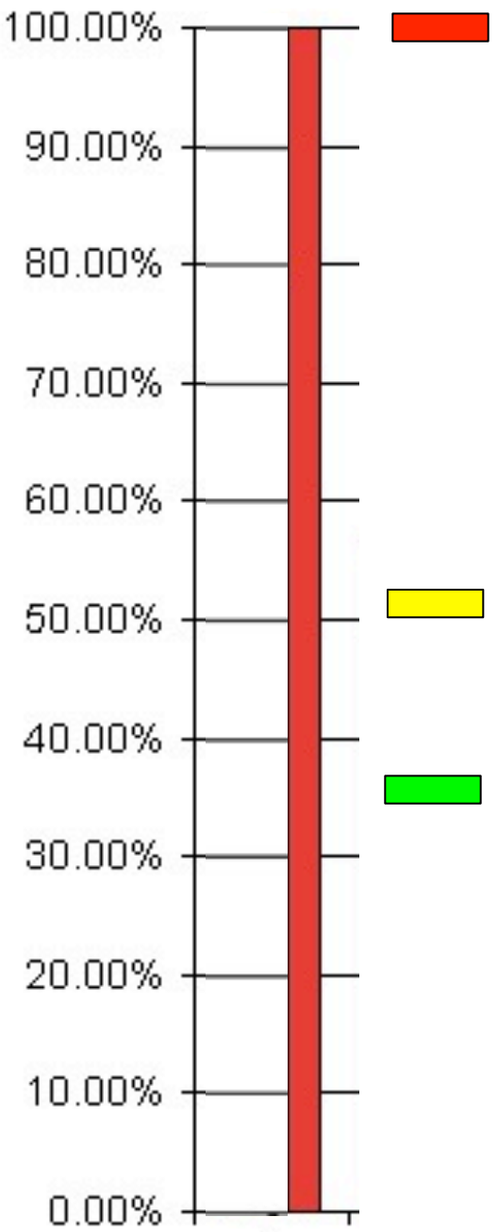
- Arithmetic throughput is a **per core resource** that scale with number of cores and frequency
- Memory bandwidth is a **shared resource between cores** on a socket

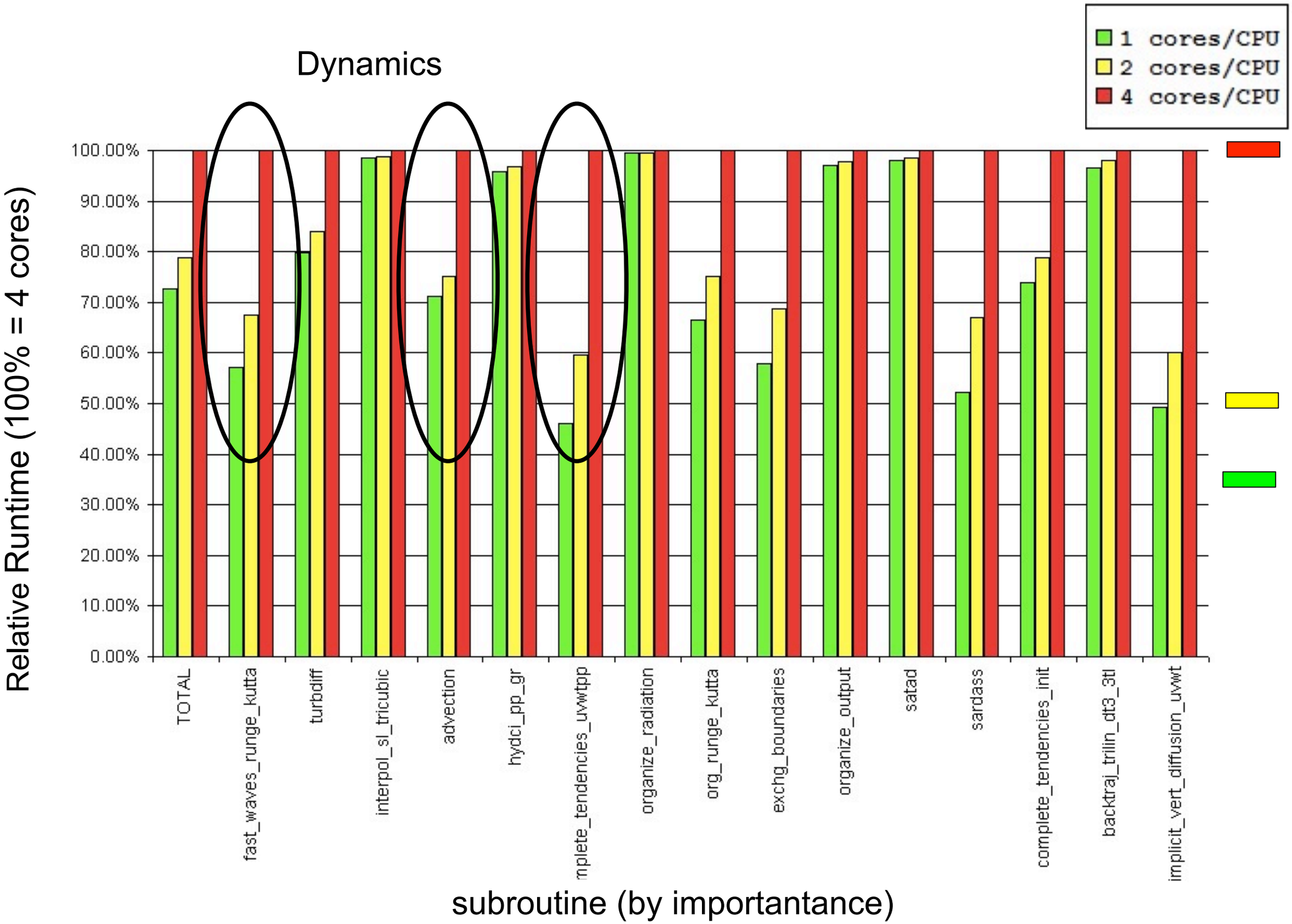
Looking at the entire COSMO code

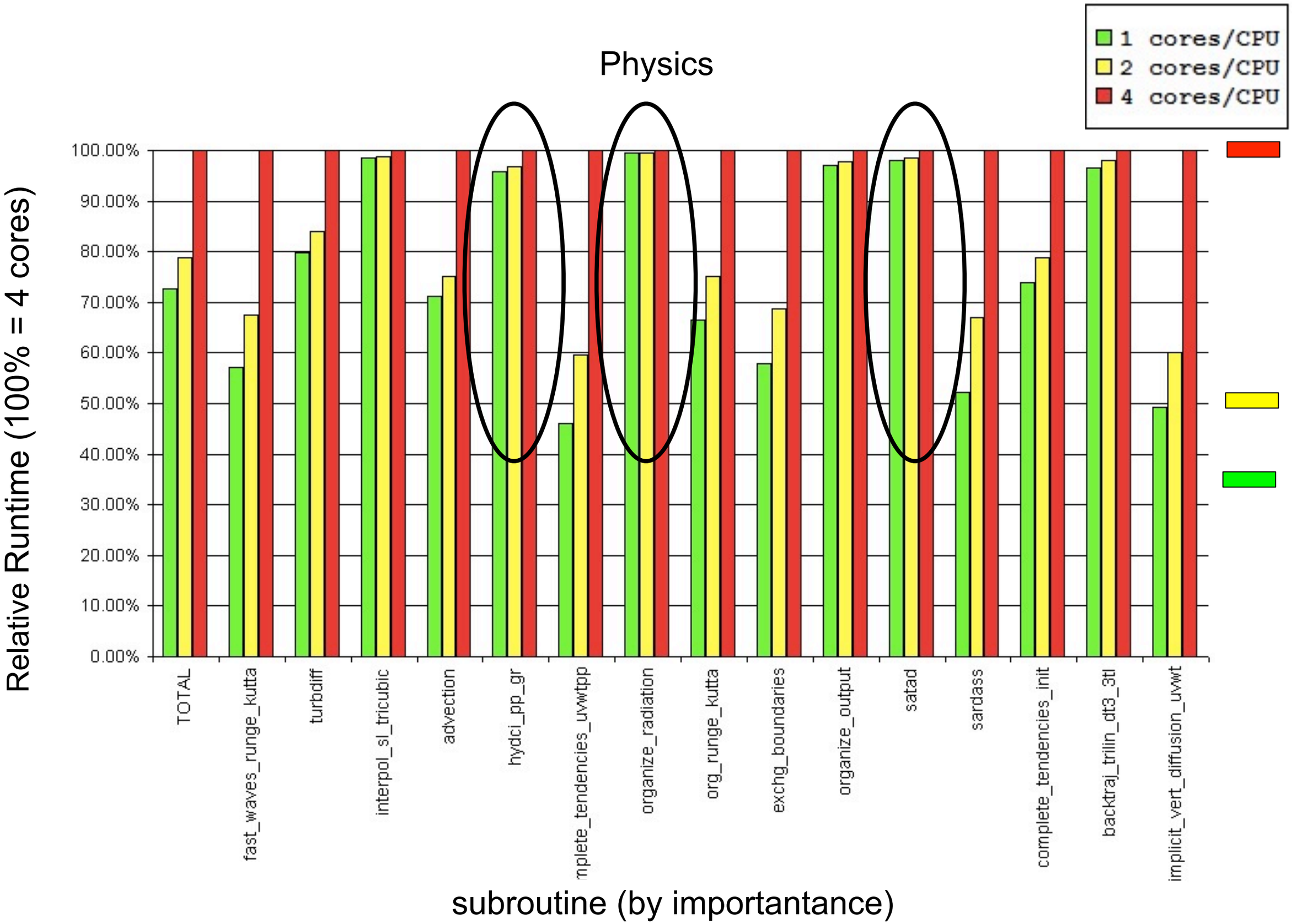


- Running COSMO-2, keeping compute power constant, but varying the available memory bandwidth
- Keep number of cores constant and vary number of cores used per socket






Dynamics
1 cores/CPU
2 cores/CPU
4 cores/CPU
Relative Runtime (100% = 4 cores)
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
TOTAL
fast_waves_runge_kutta
turbdiff
interpol_sl_tricubic
advection
hydci_pp_gr
mplete_tendencies_uvwtpp
organize_radiation
org_runge_kutta
exchg_boundaries
organize_output
satad
sardass
complete_tendencies_init
backtraj_trilin_dt3_3tl
implicit_vert_diffusion_uvwt
subroutine (by importantance)


Physics
1 cores/CPU
2 cores/CPU
4 cores/CPU
Relative Runtime (100% = 4 cores)
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
TOTAL
fast_waves_runge_kutta
turbdiff
interpol_sl_tricubic
advection
hydci_pp_gr
mplete_tendencies_uvwtpp
organize_radiation
org_runge_kutta
exchg_boundaries
organize_output
satad
sardass
complete_tendencies_init
backtraj_trilin_dt3_3tl
implicit_vert_diffusion_uvwt
subroutine (by importantance)

Running the simple examples on the Cray XK6

Compute bound (physics) problem

Machine	Interlagos	Fermi (2090)	GPU+transfer
Time	1.31 s	0.17 s	1.9 s
Speedup	**1.0 (REF)**	**7.6**	**0.7**

Memory bound (dynamics) problem

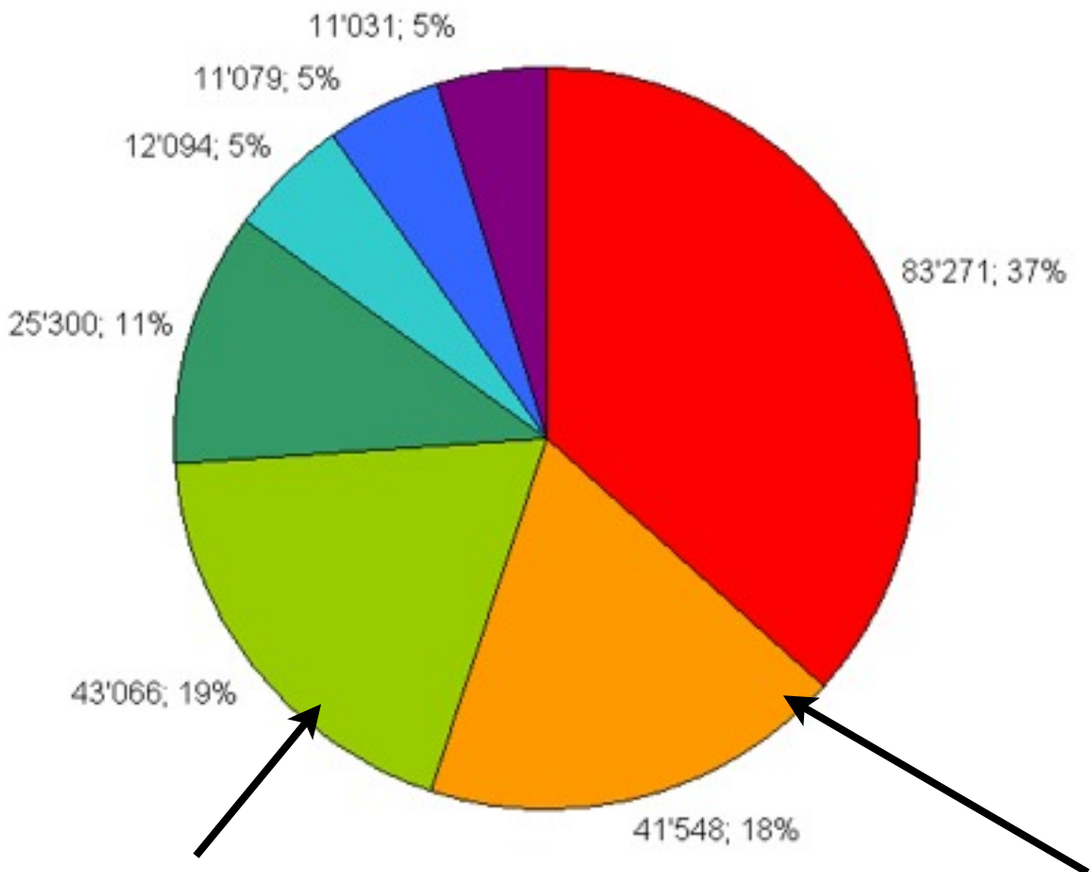
Machine	Interlagos	Fermi (2090)	GPU+transfer
Time	0.16 s	0.038 s	1.7 s
Speedup	**1.0 (REF)**	**4.2**	**0.1**

The simple lesson: leave data on the GPU!

Performance profile of (original) COSMO-CCLM

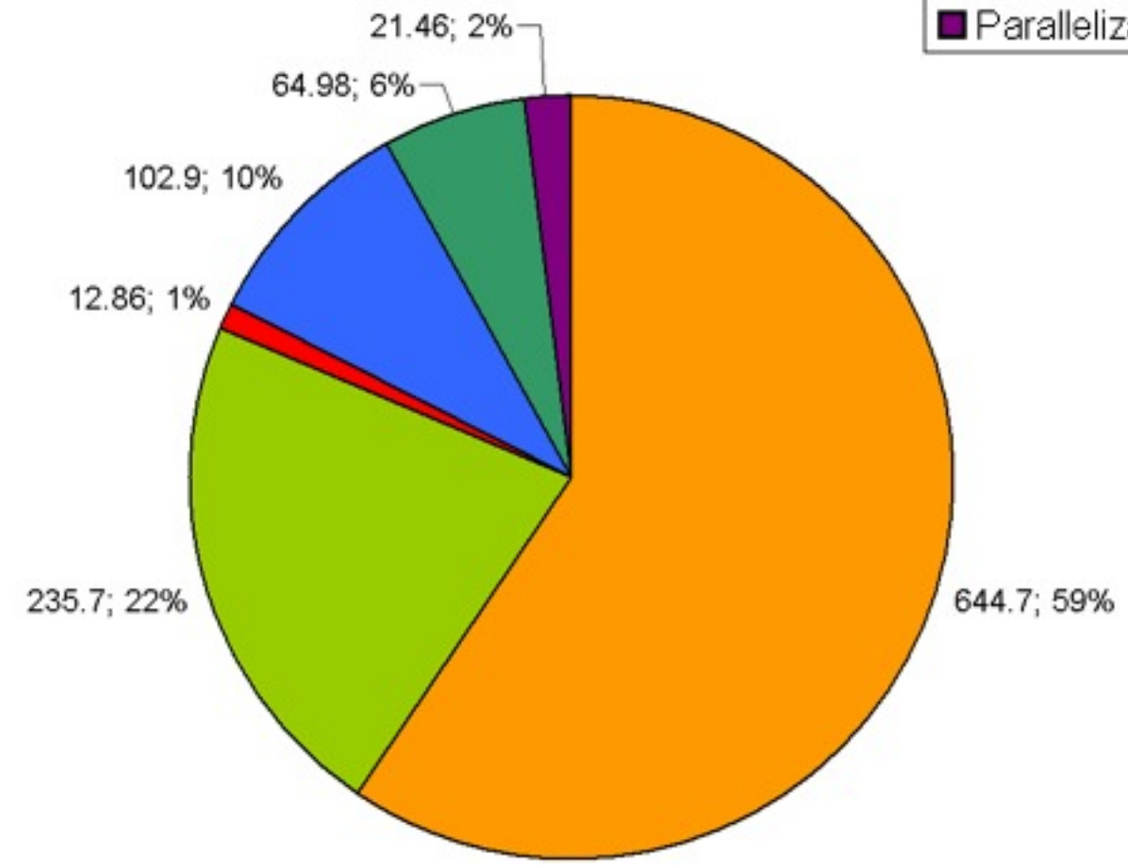
Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90)



Original code (with OpenACC)

% Runtime



Rewrite in C++ (with CUDA)

Approach to refactoring of COSMO

- Physics
  - 40k lines, 20% of runtime
  - Many developers
  - “Plug-ins” from other models
  - More compute bound

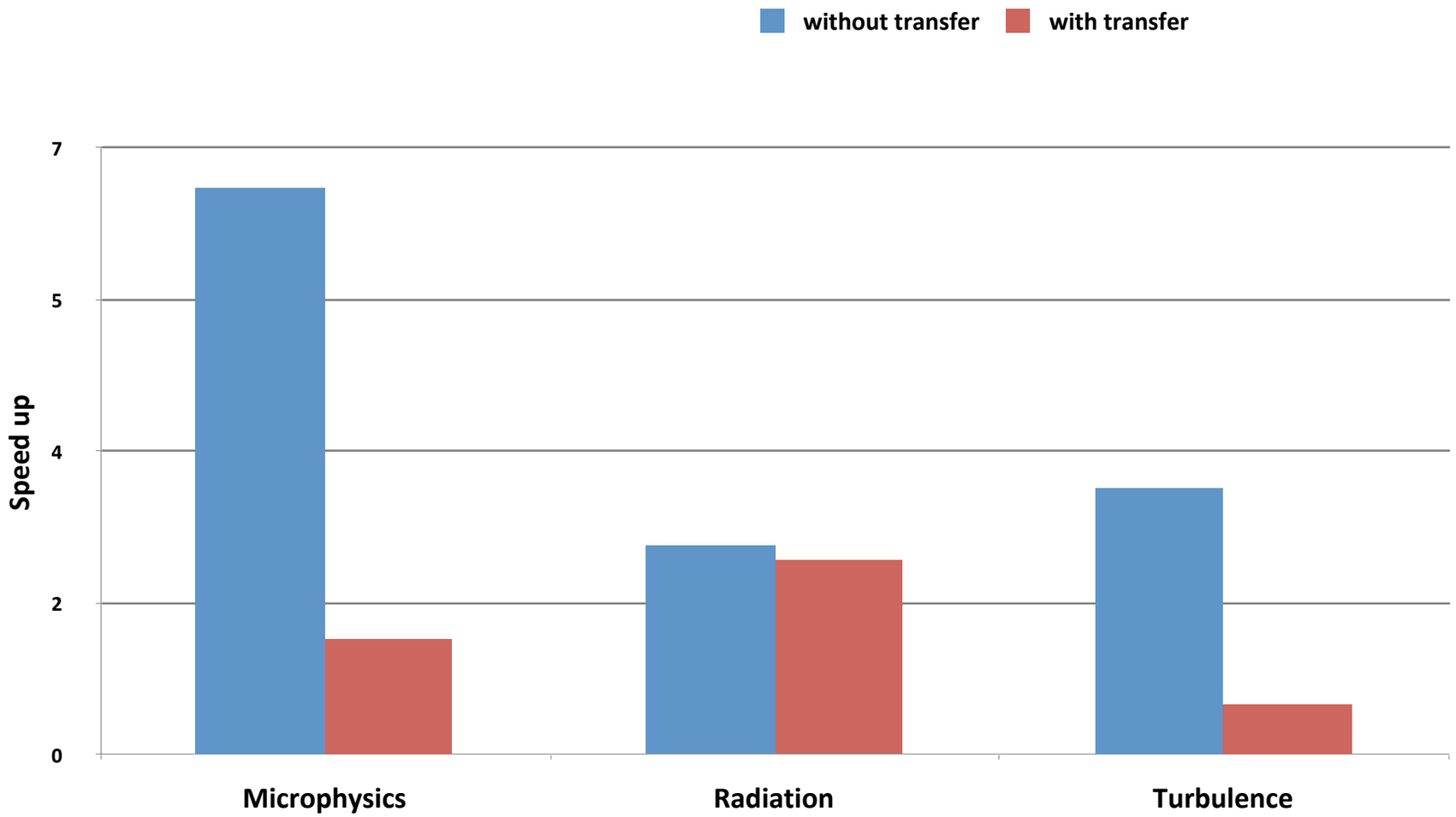
Port to GPU
- keep source
- directives

Refactoring Physics in COSMO: OpenACC directives

```
!$acc update device(a,b)
do j=1,niter
 !$acc region do kernel
 do i=1,n
  c(i) = a(i) + b(i) * ( a(i+1) - 2.0d0*a(i) + a(i-1) )
 end do
 !$acc update host(c)
end do
!$acc update host(c)
```

Unfortunately, things are not always that easy and code restructuring is required. But this usually helps improving performance if the code ran on a CPU.

Speedup GPU vs. CPU



Approach to refactoring of COSMO

- Physics
  - 40k lines, 20% of runtime
  - Many developers
  - “Plug-ins” from other models
  - More compute bound

Port to GPU
- keep source
- directives

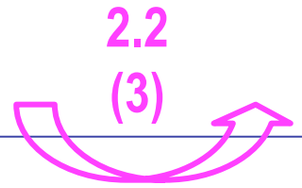
- Dynamics
  - 20k lines, 60% of runtime
  - Few developers
  - Strongly memory bandwidth bound

Aggressive rewrite
- Data structures
- C++
- DSEL

Results for dynamical core

- Fully implemented / verified agains original code
- Speedup CPU (Magny Cours) vs. GPU (Tesla C2075)

Original CPU	Rewrite CPU	Rewrite GPU
1.0	**1.8**	**4.0**



- CPU speedup du to reduction in memory access
- Further GPU optimization work is ongoing

Dynamics in COSMO-CCLM

velocities

$$\frac{\partial u}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\frac{\partial E_h}{\partial \lambda} - vV_a\right\} - \dot{\zeta}\frac{\partial u}{\partial \zeta} - \frac{1}{\rho a\cos\varphi}\left(\frac{\partial p'}{\partial \lambda} - \frac{1}{\sqrt{\gamma}}\frac{\partial p_0}{\partial \lambda}\frac{\partial p'}{\partial \zeta}\right) + M_u$$

$$\frac{\partial v}{\partial t} = -\left\{\frac{1}{a}\frac{\partial E_h}{\partial \varphi} + uV_a\right\} - \dot{\zeta}\frac{\partial v}{\partial \zeta} - \frac{1}{\rho a}\left(\frac{\partial p'}{\partial \varphi} - \frac{1}{\sqrt{\gamma}}\frac{\partial p_0}{\partial \varphi}\frac{\partial p'}{\partial \zeta}\right) + M_v$$

$$\frac{\partial w}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial w}{\partial \lambda} + v\cos\varphi\frac{\partial w}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial w}{\partial \zeta} + \frac{g}{\sqrt{\gamma}}\frac{\rho_0}{\rho}\frac{\partial p'}{\partial \zeta} + M_w + g\frac{\rho_0}{\rho}\left\{\frac{(T-T_0)}{T} - \frac{T_0 p'}{T p_0} + \left(\frac{R_v}{R_d} - 1\right)q^v - q^l - q^f\right\}$$

pressure

$$\frac{\partial p'}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial p'}{\partial \lambda} + v\cos\varphi\frac{\partial p'}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial p'}{\partial \zeta} + g\rho_0 w - \frac{c_{pd}}{c_{vd}}pD$$

temperature

$$\frac{\partial T}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial T}{\partial \lambda} + v\cos\varphi\frac{\partial T}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial T}{\partial \zeta} - \frac{1}{\rho c_{vd}}pD + Q_T$$

water

$$\frac{\partial q^v}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial q^v}{\partial \lambda} + v\cos\varphi\frac{\partial q^v}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial q^v}{\partial \zeta} - (S^l + S^f) + M_{q^v}$$

$$\frac{\partial q^{l,f}}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial q^{l,f}}{\partial \lambda} + v\cos\varphi\frac{\partial q^{l,f}}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial q^{l,f}}{\partial \zeta} - \frac{g}{\sqrt{\gamma}}\frac{\rho_0}{\rho}\frac{\partial P_{l,f}}{\partial \zeta} + S^{l,f} + M_{q^{l,f}}$$

turbulence

$$\frac{\partial e_t}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial e_t}{\partial \lambda} + v\cos\varphi\frac{\partial e_t}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial e_t}{\partial \zeta} + K_m^v\frac{g\rho_0}{\sqrt{\gamma}}\left\{\left(\frac{\partial u}{\partial \zeta}\right)^2 + \left(\frac{\partial v}{\partial \zeta}\right)^2\right\} + \frac{g}{\rho\theta_v}F^{\theta_v} - \frac{\sqrt{2}e_t^{3/2}}{\alpha_M l} + M_{e_t}$$

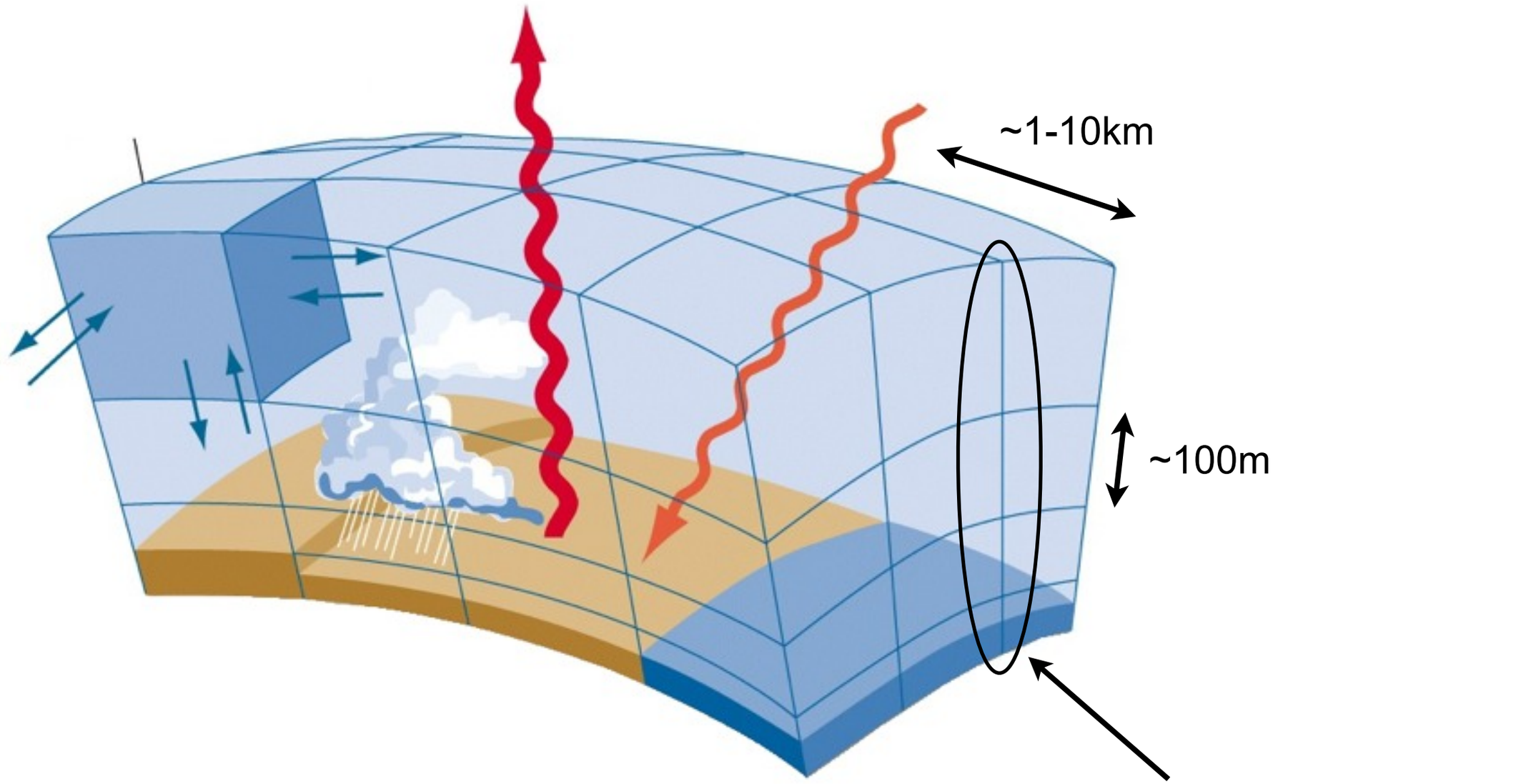
Timestep

implicit (sparse) — explicit (RK3) — implicit (sparse solver) — explicit (leapfrog)

1x physics et al. tendencies

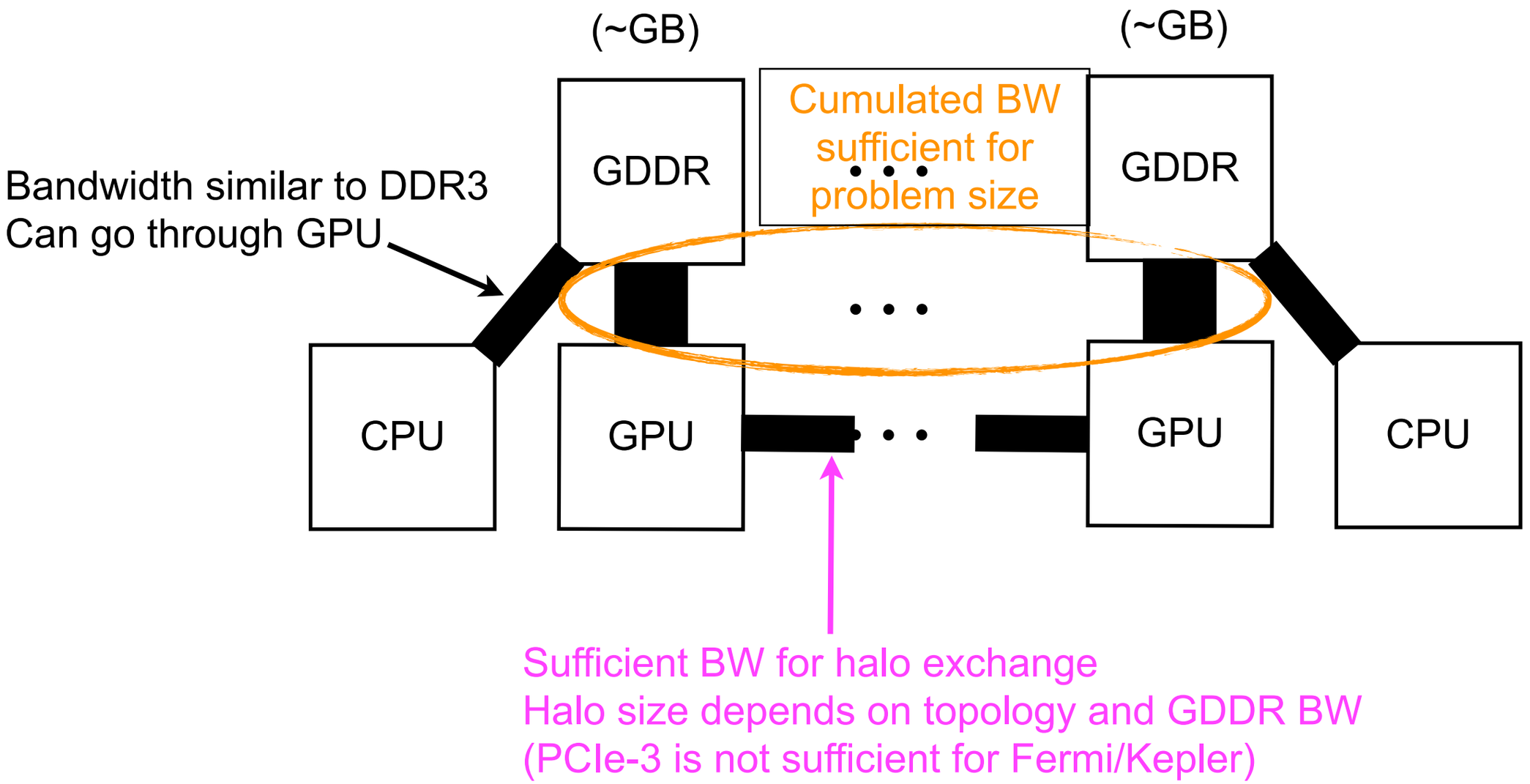
3x horizontal adv. — vertical adv. — ~10x fast wave solver

water adv.

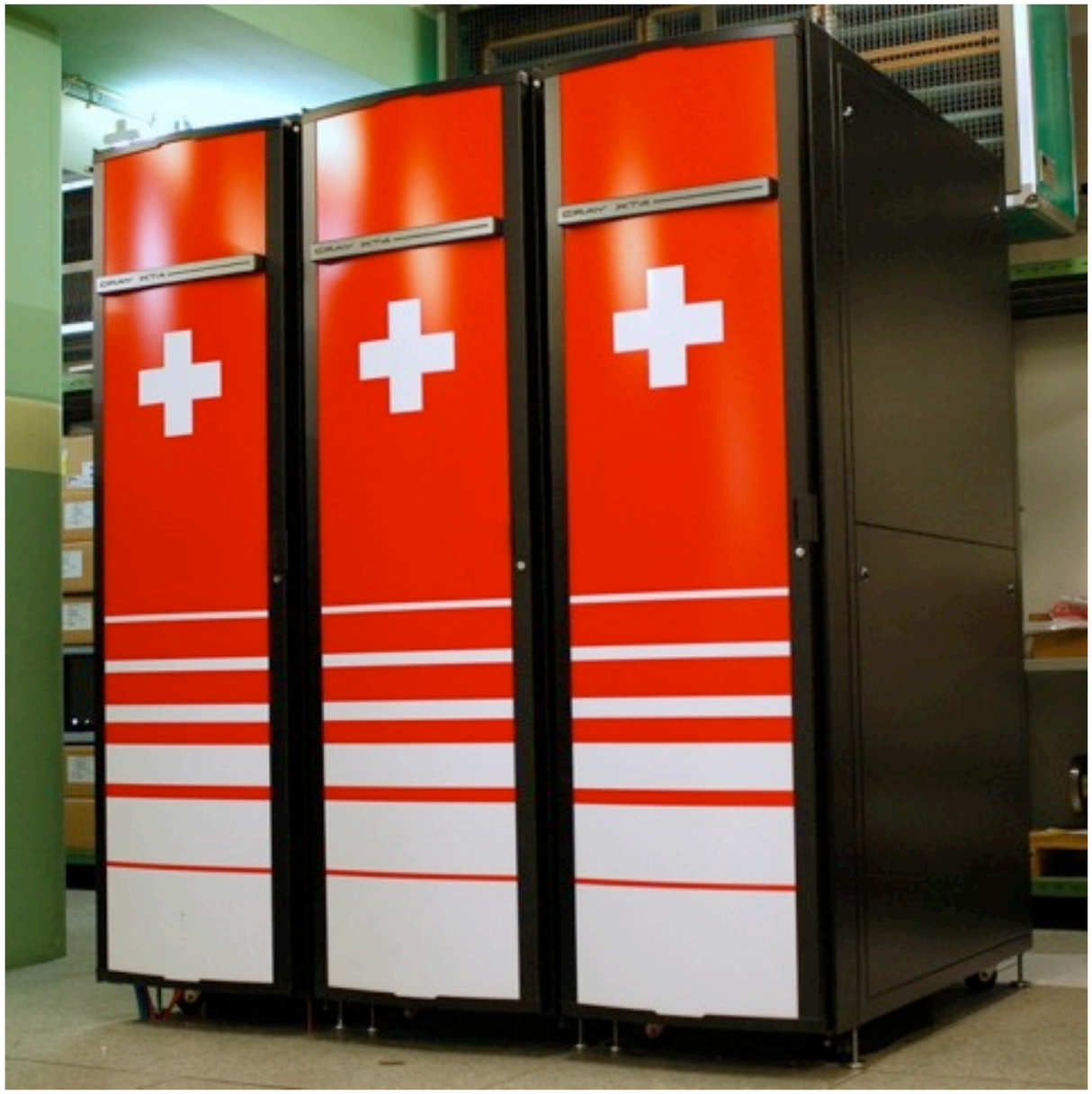


Current status of implementation:
> structure grid motif works well on data parallel, high-memory bandwidth “accelerator”
> tridiagonal solve fits in L2 cash on CPU and makes good use of single thread performance

A crude way to consider the architecture we need for physics and dynamics in COSMO (preliminary)

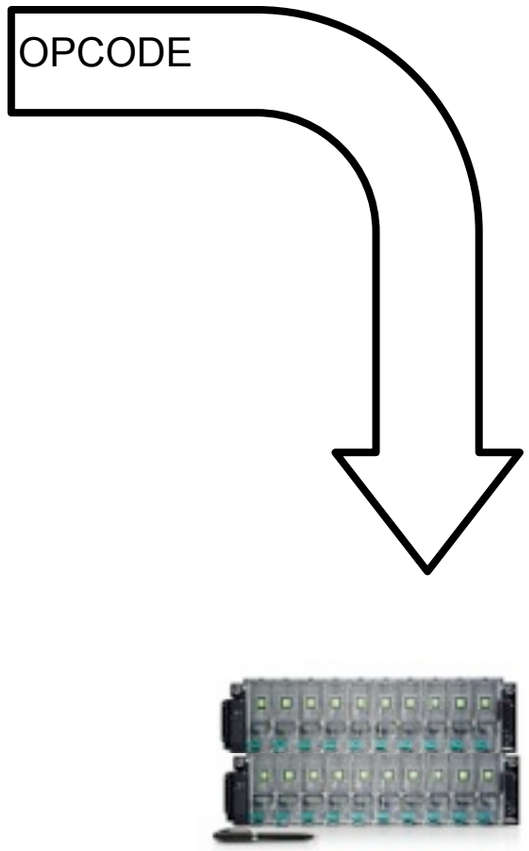


OPCODE project: can we run entire MeteoCH production suit on a node with ~8-16 GPUs?



Cray XT4 (Production Machine at CSCS)

246 AMD Opteron Barcelona

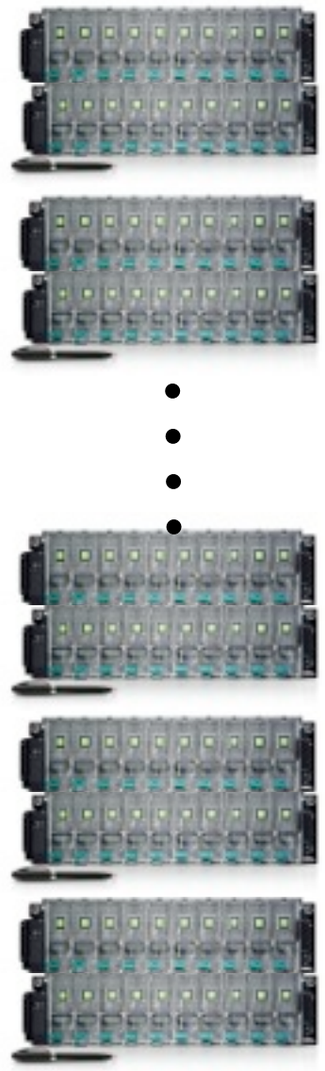


Server

O(20) GPUs

Many such Server

for ensemble runs



CONCLUSIONS

We can have both, higher resolution and larger ensembles!

But we have to invest in code refactoring and algorithm re-engineering in order to consider appropriate architectures for our problem*

(*) applies to all 12 projects we are studying in the HP2C platform (see www.hp2c.ch)

THANK YOU!