ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

Why hybrid systems work so well for simulation based sciences

Thomas C. Schulthess

BECAUSE OF AMDAHL'S LAW!

We knew this all along, so this is trivial

But how exactly does this show in applications?

We have 20 minutes to look at two applications (that are taken from real life simulations)

Electronic structure problem (DFT-based) in material science

Collaborators: Stan Tomov[1], Azam Haidar[1], Raffaele Solcà[2], Antron Koszevnikov[2], Jack Dongarra[1]

Regional model for climate simulations and weather forecasting

Collaborators: Oliver Fuhrer[3], Tobias Gysi[4], Will Sawyer[5], David Müller[4], Uli Schätler[6], Michael Baldauf[6], Xavier Lapillone[2], Ugo Vareto[5], Mauro Bianco[5], Isabelle Bey[2], Tim Schröder[7], Tom Bradley[7]

Many insightful discussion with Steve Scot[7], John Levesque[8], and Alessandro Curioni[9] (and many others)

(1) ICL/UTK; (2) ETH; (3) MeteoCH; (4) SCS; (5) CSCS; (6) DWD; (7) NVIDIA; (8) Cray; (9) IBM-ZRL

Solving Kohn-Sham equation is the bottleneck of most DFT based materials science codes

Kohn-Sham Eqn. $$\left(-\frac{\hbar^2}{2m}\nabla^2 + v_{\mathrm{LDA}}(\vec{r})\right)\psi_i(\vec{r}) = \epsilon_i\psi_i(\vec{r})$$

Ansatz $$\psi_i(\vec{r}) = \sum_\mu c_{i\mu}\phi_\mu(\vec{r})$$

Hermitian matrix $$H_{\mu\nu} = \int \phi^*_\mu(\vec{r})\left(-\frac{\hbar^2}{2m}\nabla^2 + v_{\mathrm{LDA}}(\vec{r})\right)\phi_\nu(\vec{r})d\vec{r}$$

Basis is not orthogonal $$S_{\mu\nu} = \int \phi^*_\mu(\vec{r})\phi_\nu(\vec{r})d\vec{r}$$

Solve generalized eigenvalue problem $$(\mathbf{H} - \varepsilon_i\mathbf{S}) = 0$$

where we are usually interested in about 10-50% of spectrum

We need eigenvectors as well, to compute the density:

$$n(\xi) = \sum_{i=1}^{N}\psi^*_i(\xi)\psi_i(\xi)$$

Solving the generalized eigenvalue problem

$$\mathbf{Ax} = \lambda\mathbf{Bx}$$

$$\mathbf{B} = \mathbf{LL}^H$$

$$\mathbf{A}' = \mathbf{L}^{-1}\mathbf{AL}^{-H}$$

$$\mathbf{A}'\mathbf{y} = \lambda\mathbf{y}$$

$$\mathbf{x} = \mathbf{L}^{-H}\mathbf{y}$$

$$\mathbf{A}'\mathbf{y} = \lambda\mathbf{y}$$

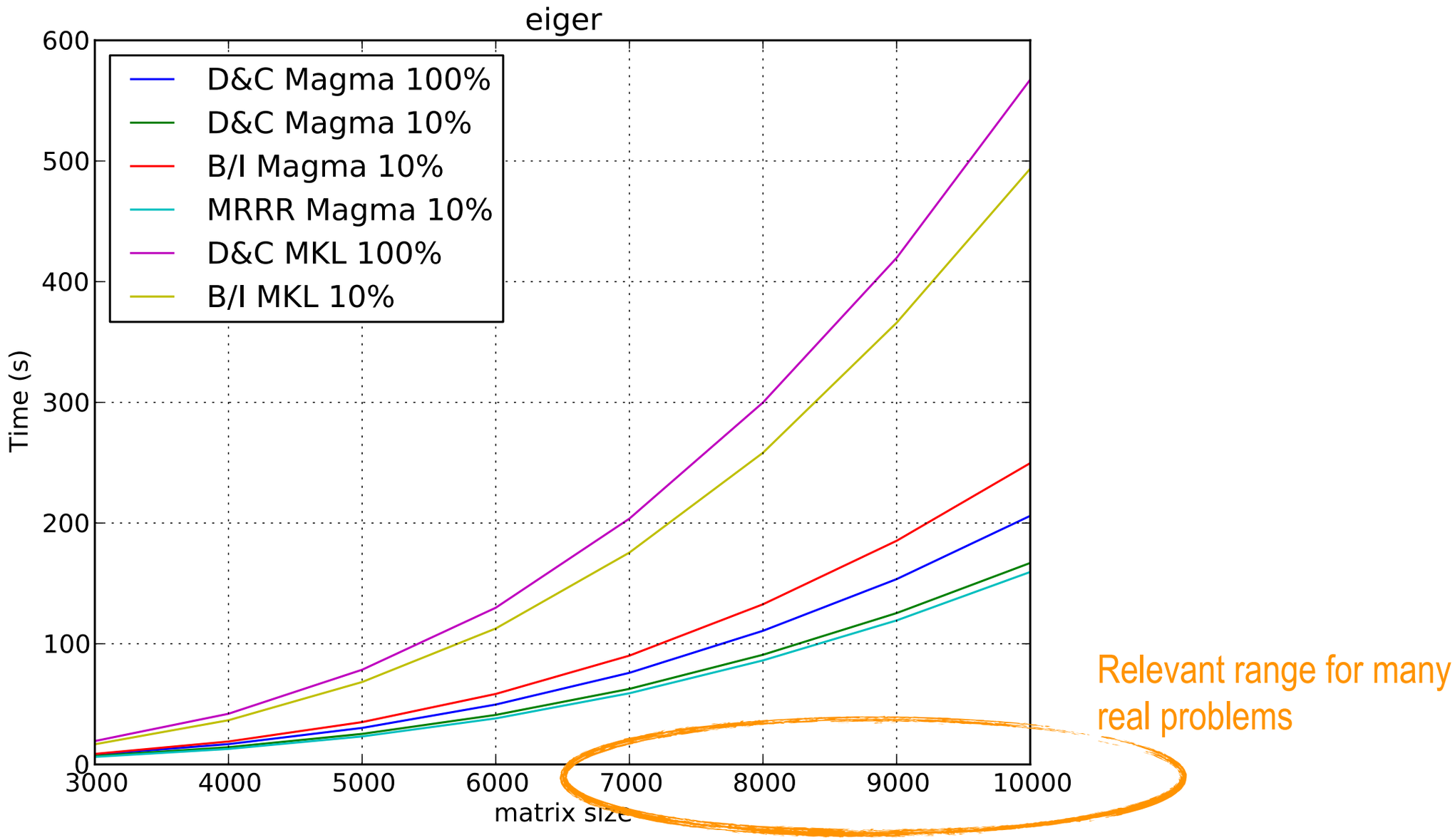
$$\mathbf{T} = \mathbf{Q}^H\mathbf{A}'\mathbf{Q}$$

$$\mathbf{Ty}' = \lambda\mathbf{y}'$$

$$\mathbf{y} = \mathbf{Qy}'$$

Comparing multi-core vs. hybrid system

Test system: two 6-core AMD Opteron 2427@2.2GHz and one Tesla C2070



Relevant range for many real problems

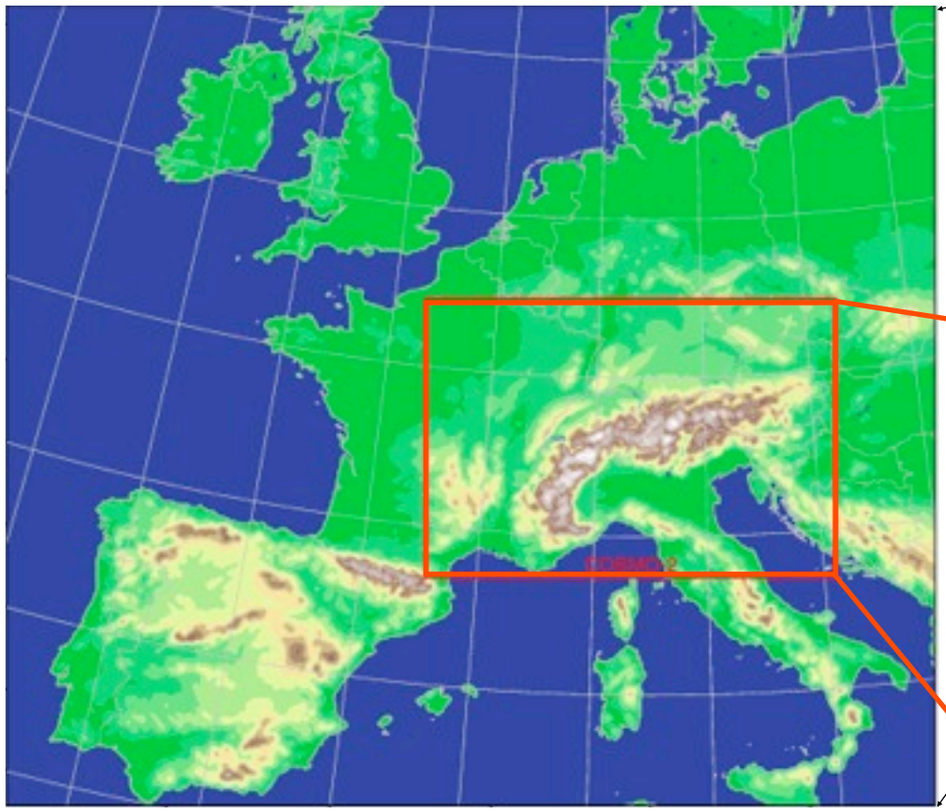
MKL runs on two AMD socket (i.e. 12 CPU cores) **MAGMA** runs on 1 CPU core and the GPU

(much care was taken to run both cases optimally)

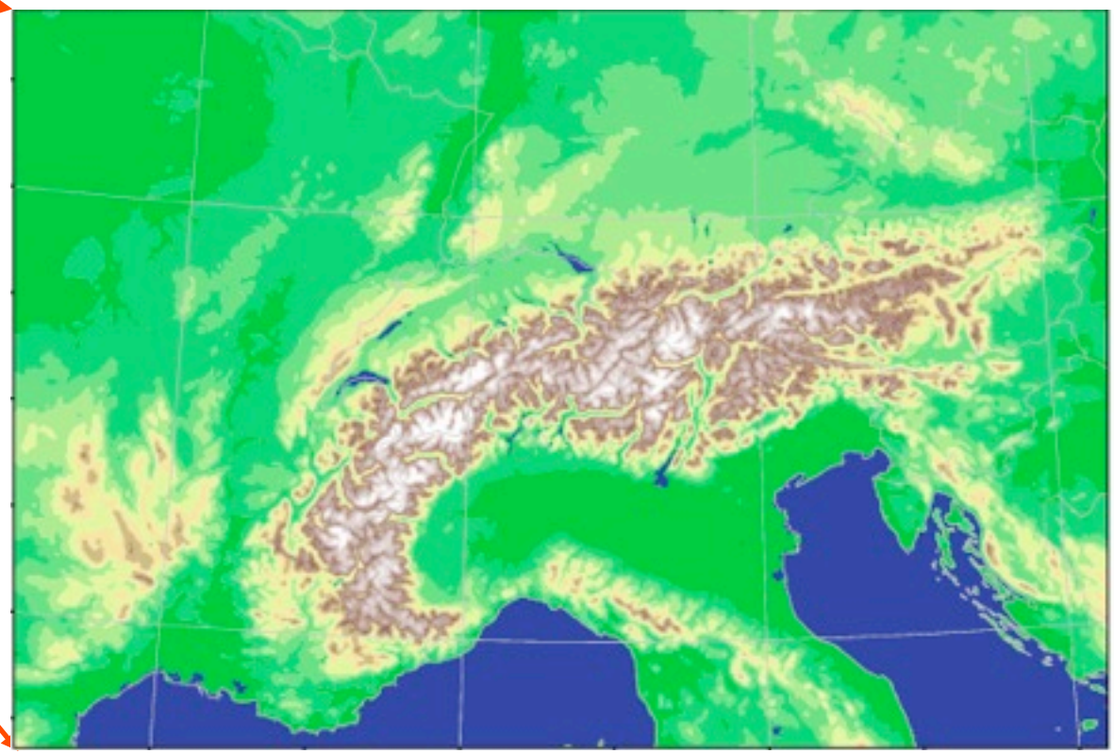
Numerical weather predictions for Switzerland (MeteoSwiss)

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

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



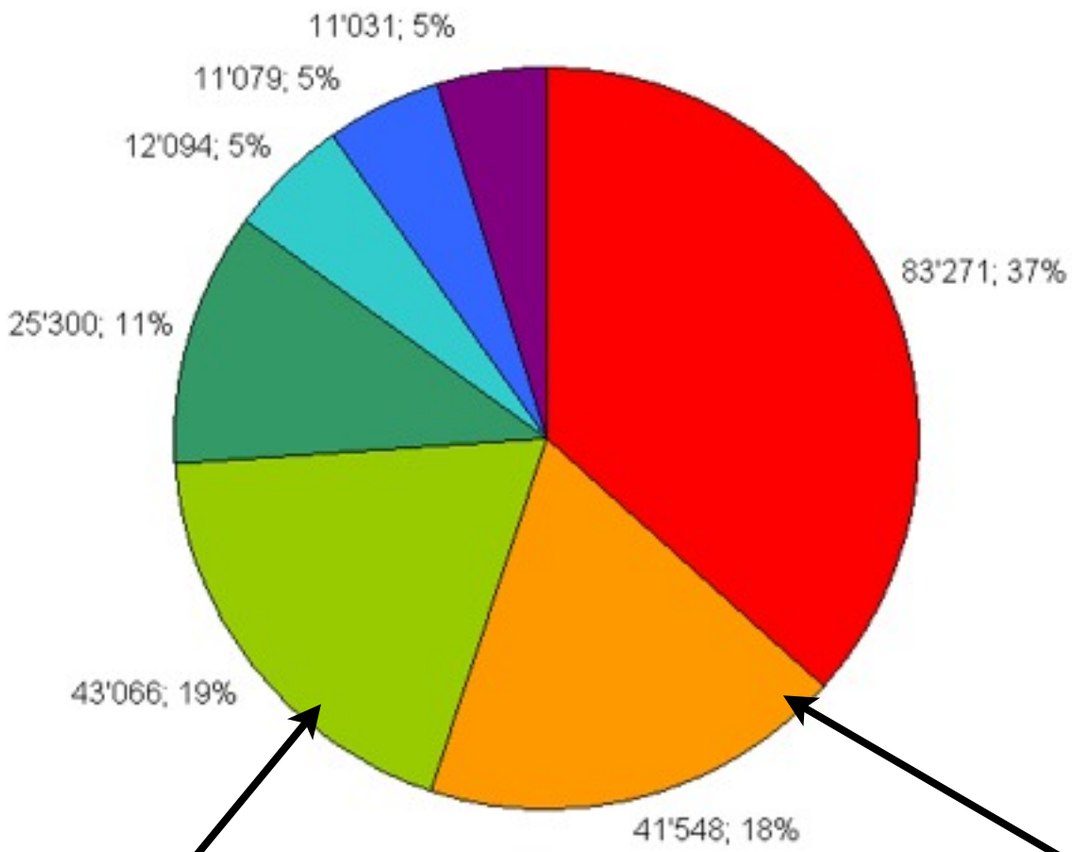
COSMO-2: 8x per day 24h forecast
2.2 km lateral grid , 60 layer



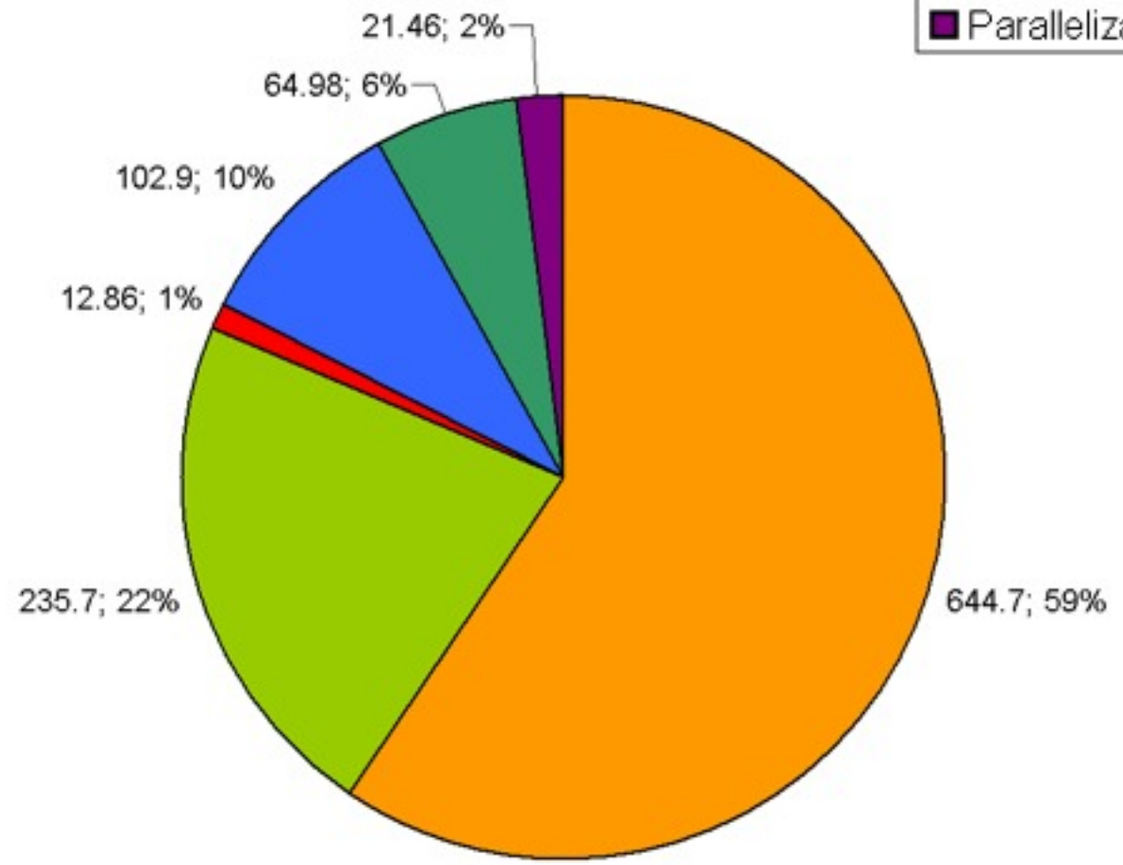
Performance profile of (original) COSMO-CCLM

Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90)



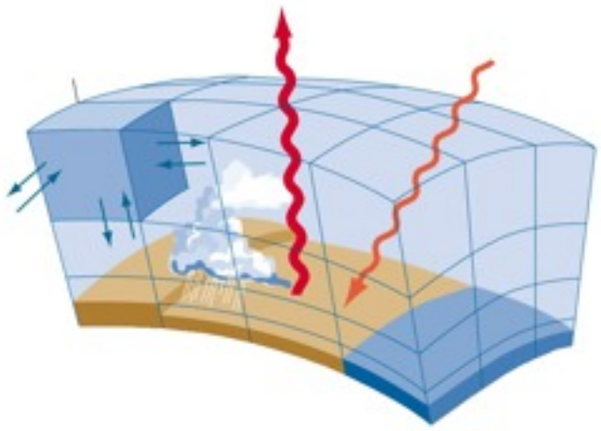
% Runtime

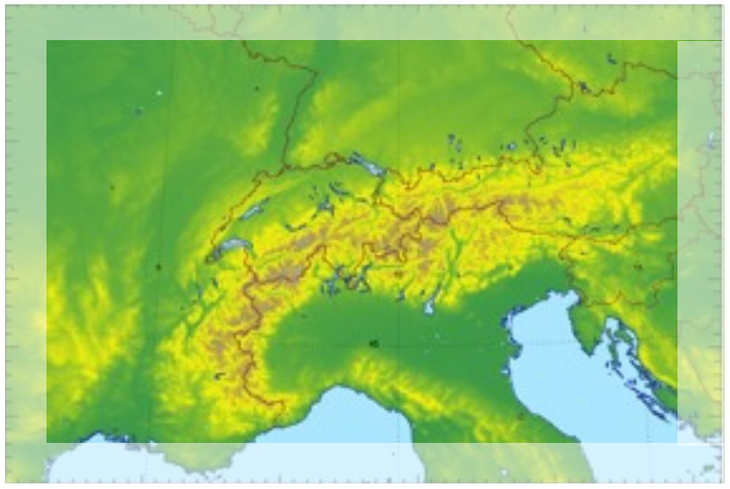


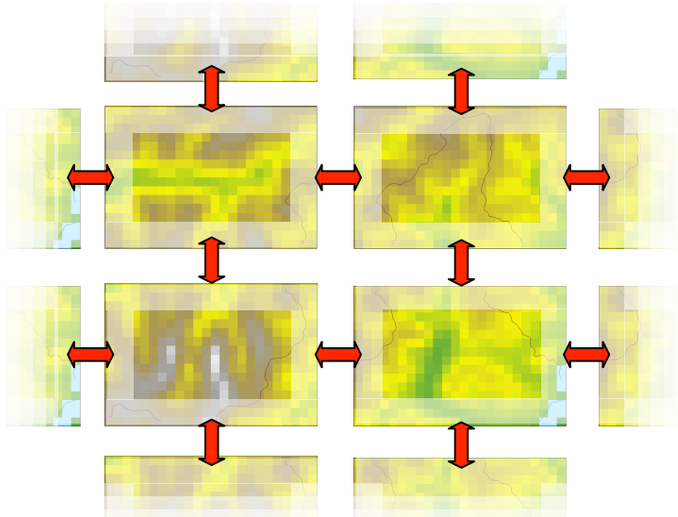
Original code (with OpenACC)

Rewrite in C++ (with CUDA)

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

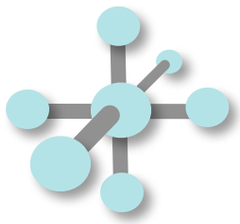


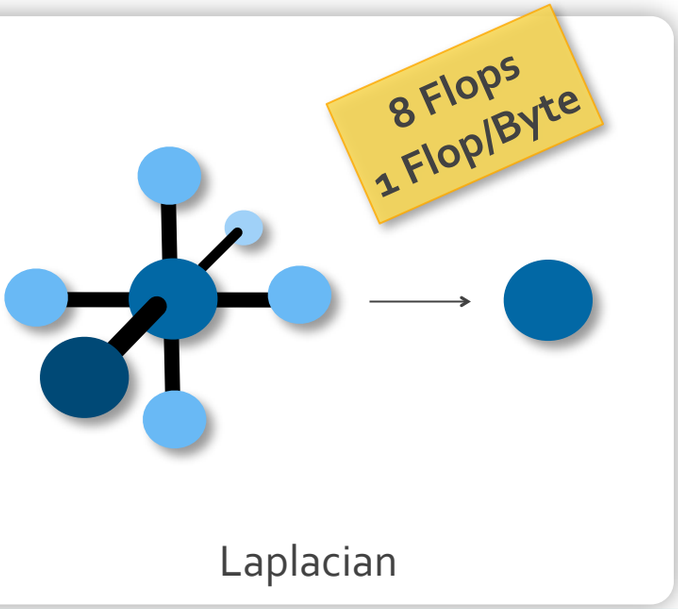


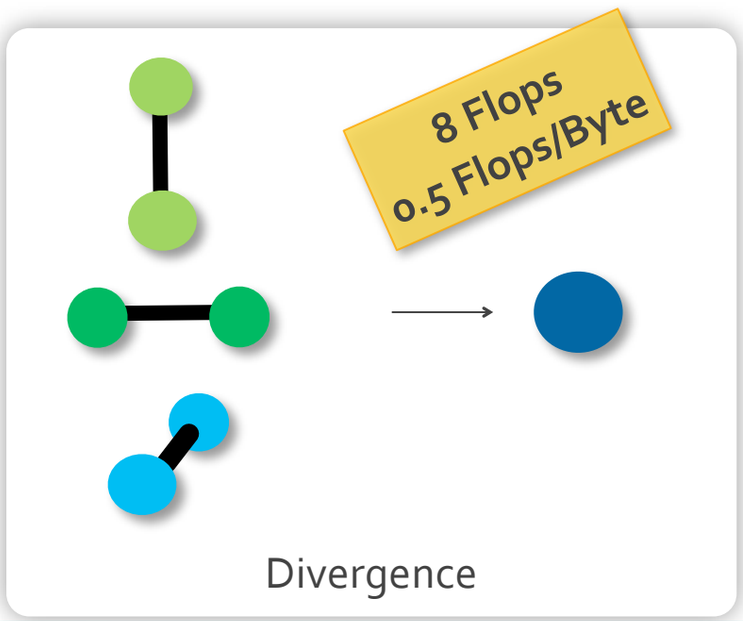


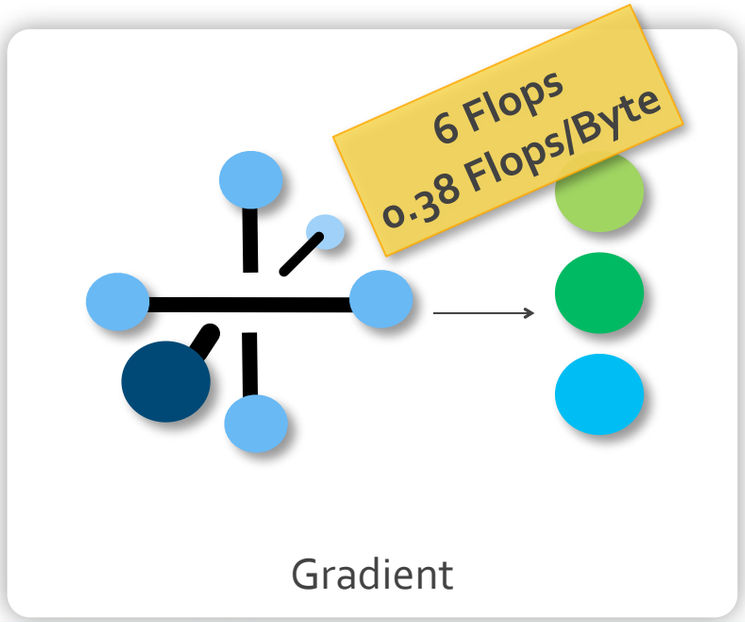
Partial differential equations on structured grid

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









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

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

water

$$\frac{\partial q^v}{\partial t}, \quad \frac{\partial q^{l,f}}{\partial t}$$

> Structured grid / stencil computation is ideal for GPU (as a bandwidth engine)

> Small memory footprint allows up to put entire problem on “accelerator(s)”

> But the tridiagonal solve is inherently a serial

turbulence

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

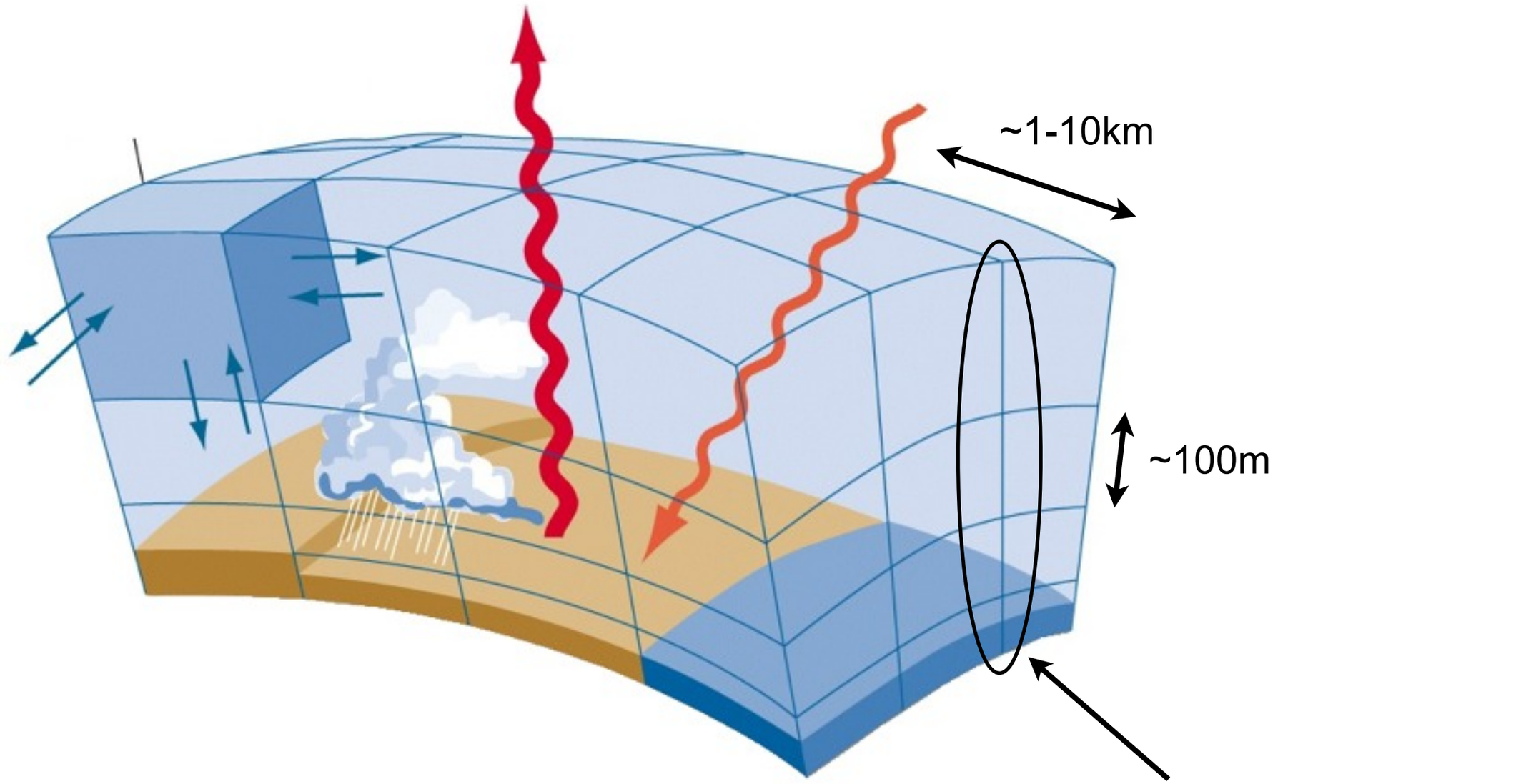
Timestep

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 makes good use of cash on CPU and single thread performance

Hybrid system we need for simulation based science has to have four “things”

- Many data parallel “high throughput” processing units
- Something that gives the highest single (several) thread performance
- Enough memory to store vectors/matrices that supports highest possible bandwidth
- Lots of memory to store everything else and supports decent bandwidth

Note: I have not talked about distributed memory systems because the possibilities are vast and I only had 20 minutes

QUESTIONS / COMMENTS