ETH zürich

Delivering performance in scientific simulations: present and future role of GPUs in supercomputers

Thomas C. Schulthess

What exactly is good performance?

What we really should care about is …

Energy & Time

But what about FLOPs and GFlops/Watt

Is Top500 / Green500 meaningless?

Metric for time to solution in High-Performance LINPACK benchmark:

(1) high arithmetic density: $\frac{\#\text{ of flop}}{\#\text{ of load-stores}} \propto O(N)$

(2) total work measured is number of retired floating point operations ($totflop$), can be easily computed

(3) normalized time to solution: $\frac{\Delta t_c}{totflop}$ or performance $\frac{totflop}{\Delta t_c}\left[\frac{\text{flop}}{\text{sec.}} = \text{flops}\right]$

… and a metric for energy to solution of HPL:

(1) normalized energy to solution E by simple measure of work $\frac{E}{totflop}$

(2) minimizing energy to solution is equivalent to maximizing $\frac{totflop}{E}\left[\frac{\text{flop}}{\text{Joule}}\right]$

(3) ... and of course $\left[\frac{\text{flop}}{\text{Joule}}\right] = \left[\frac{\frac{\text{flop}}{\text{sec.}}}{\text{Watt}}\right]$

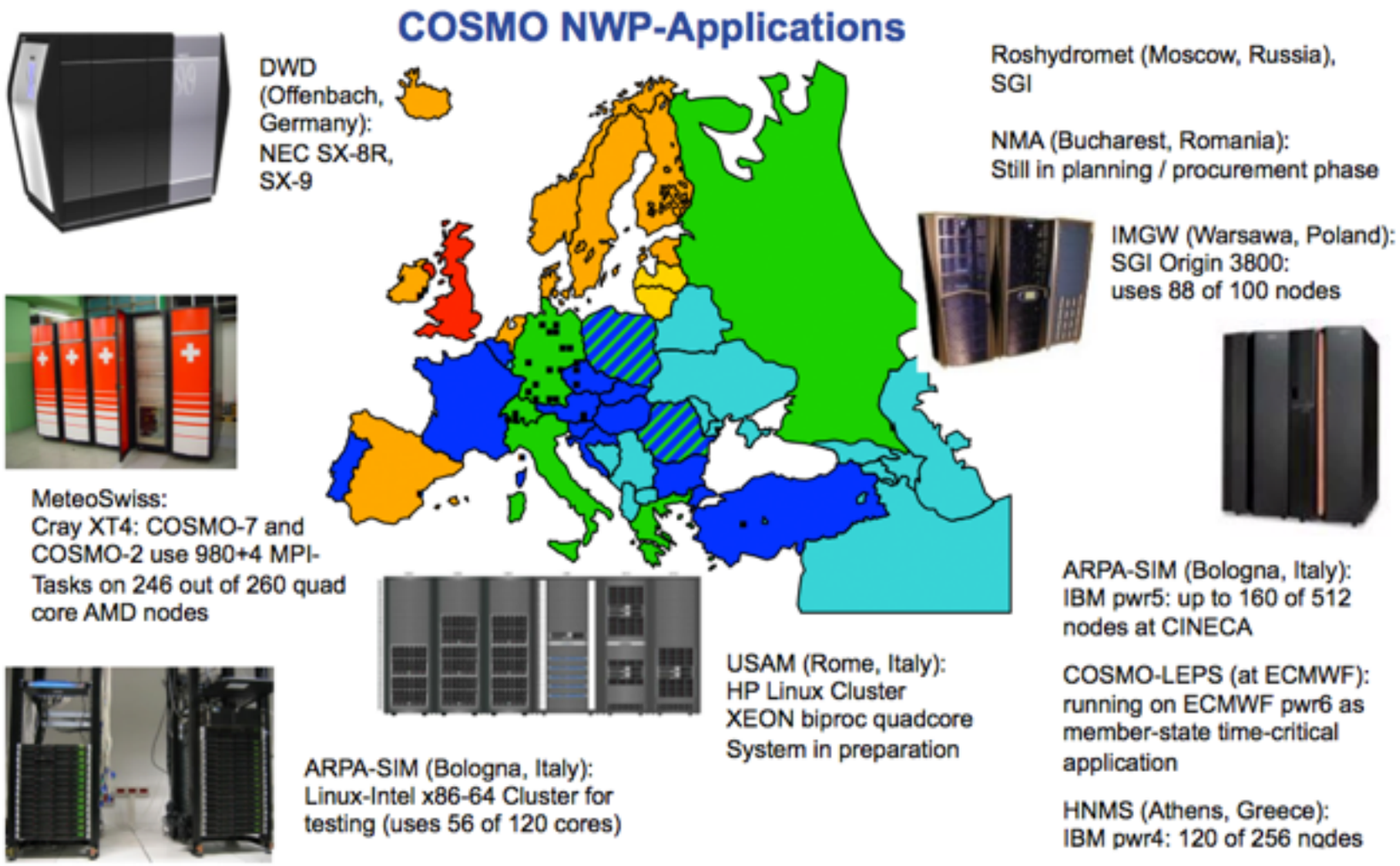
So High-Performance Linpack, Top500, and Green500 are meaningful after all

- High-Performance Linpack (HPL) or, more generally, dense linear algebra is an algorithmic motif used in many applications (e.g. quantum simulations)
- Minimising time to solution in HPL (dense linear algebra motifs) is equivalent to maximising floating point per seconds (i.e. ranking on Top500)
- Minimising energy to solution is equivalent to maximising [flop/Joule] or [Gigaflops/Watt] (i.e. ranking on Green500)

Top500 / Green500 are meaningful for certain applications

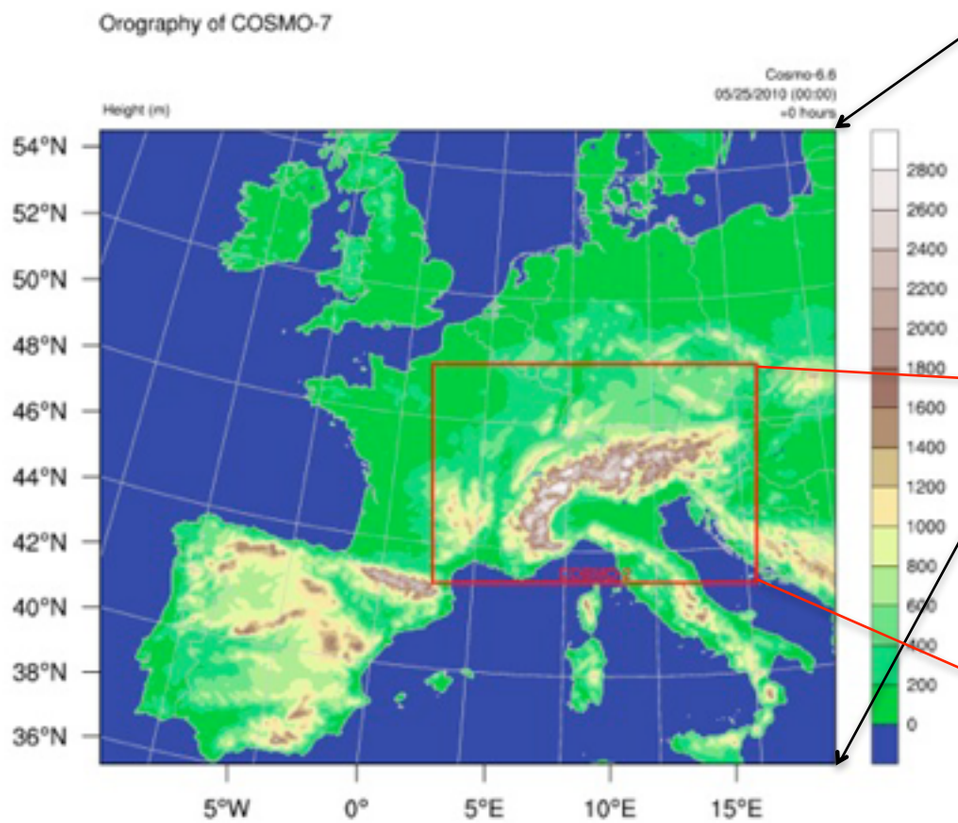
COSO: limited-area climate modelling application

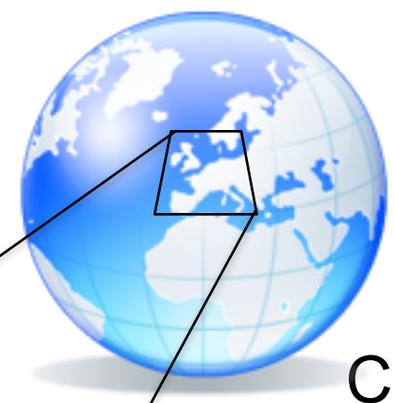
- **C**onsortium for **S**mall-**S**cale **MO**deling
- Limited-area climate model (http://www.cosmo-model.org)
- Used by 7 weather services as well as ~50 universities & research institutes



COSMO in production at Meteo Swiss

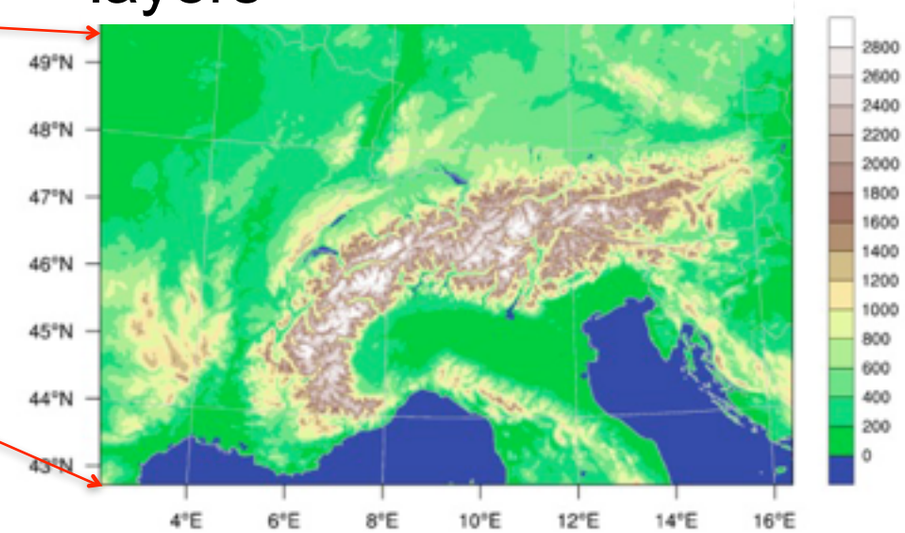
COSMO-7
3x per day 72h forecast
6.6 km lateral grid, 60 layers





ECMWF
2x per day
16 km lateral grid, 91 layers

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

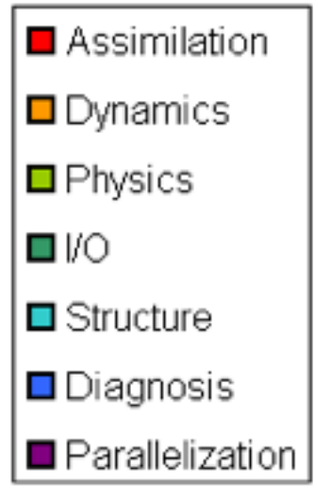


Some of the products generated with COSMO-2

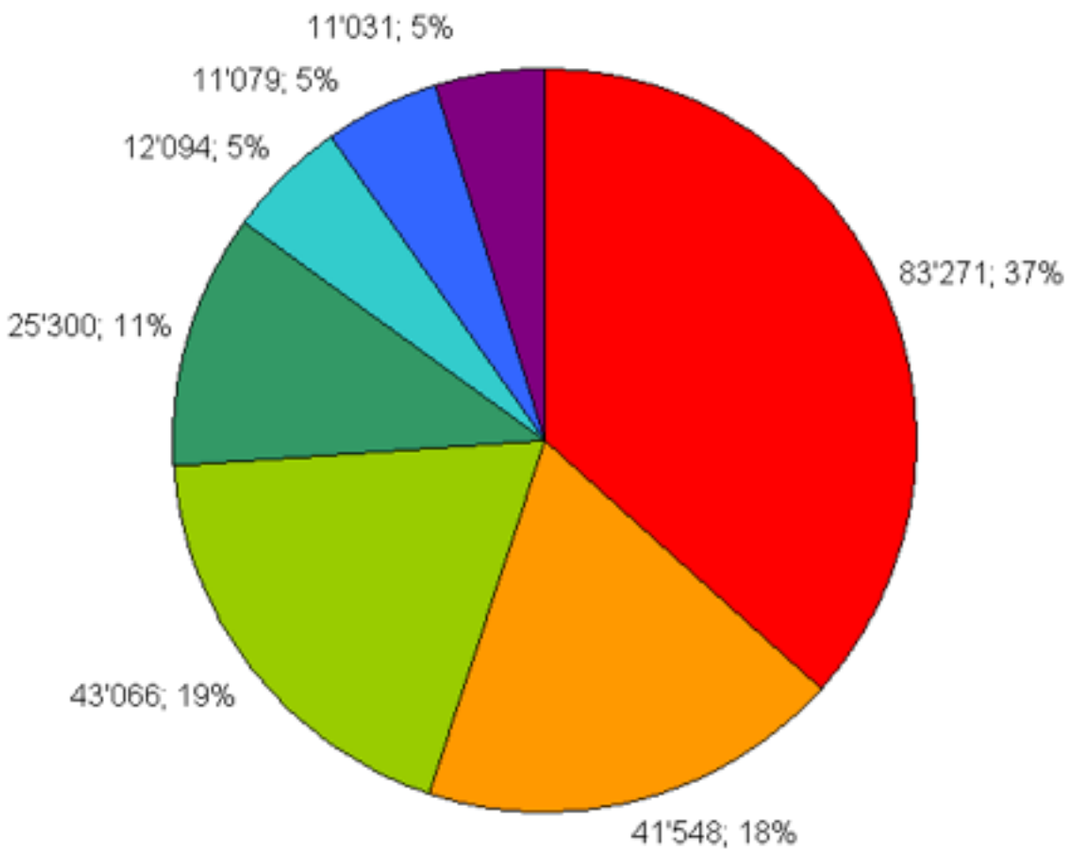
- Daily weather forecast
- Forecasting for Swiss air traffic control (Sky Gide)
- Safety management in events of nuclear incidents

Performance profile of original COSMO-2 model

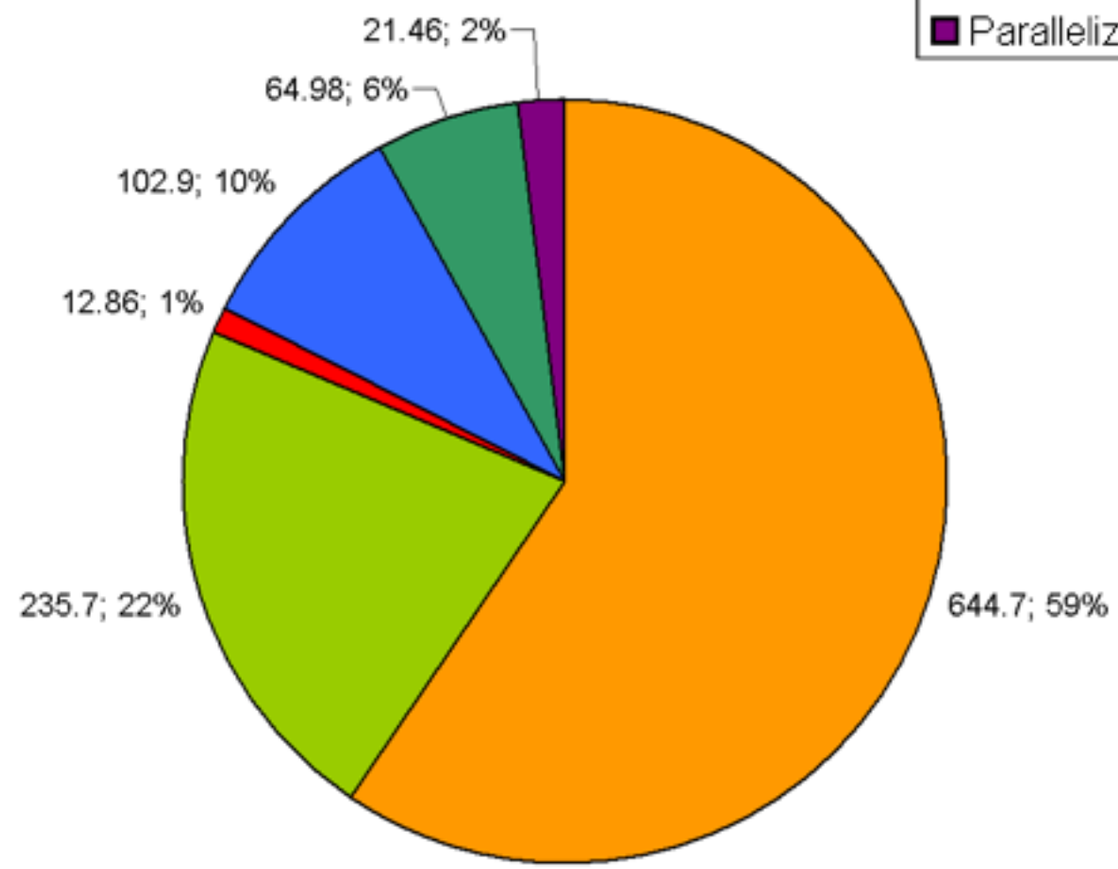
Runtime based 2 km production model of MeteoSwiss



% Code Lines (F90)



% Runtime



Typical code snippets

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

Running the snippets on a Cray XK6 (early 2012)

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**

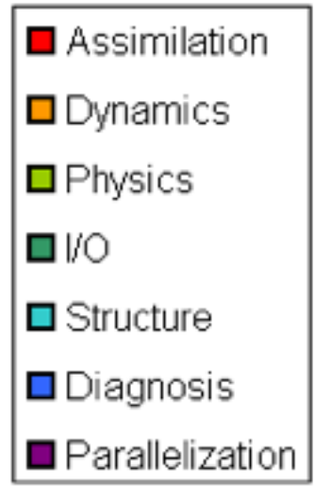
Memory bound (dynamics) problem

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

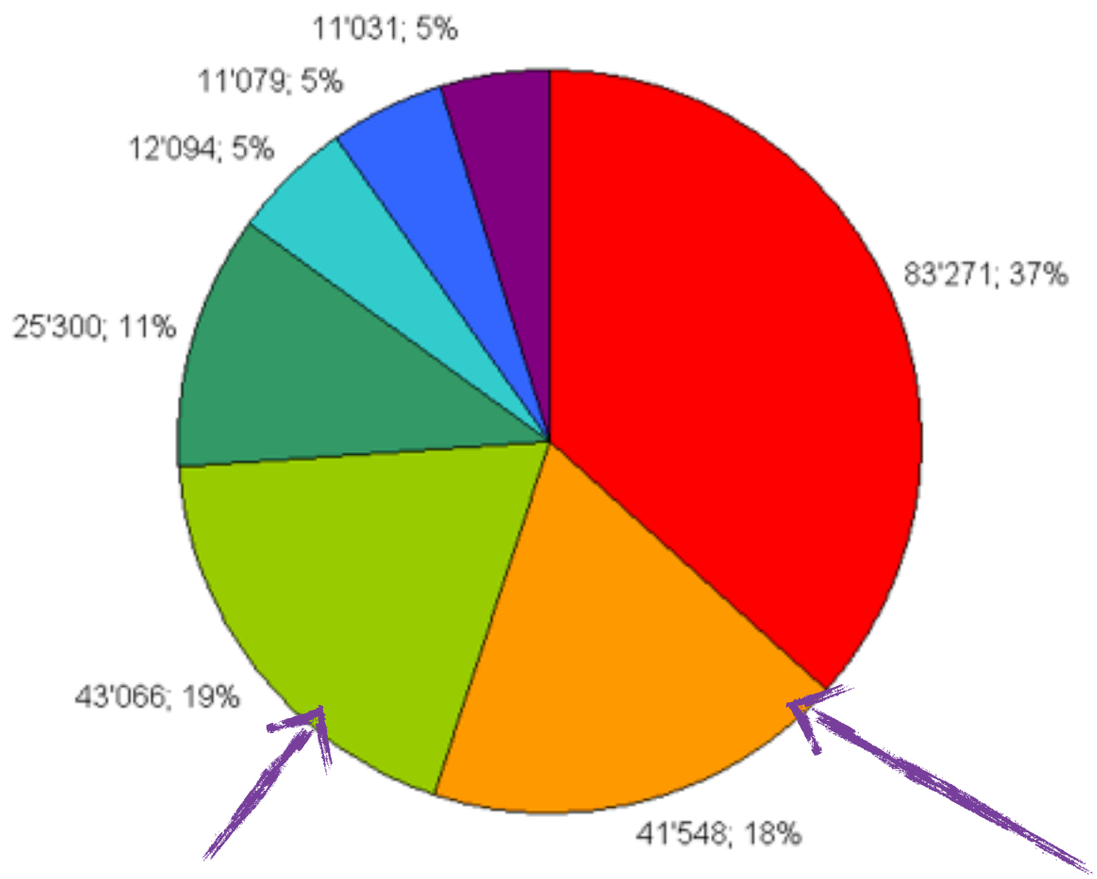
Pay attention to data management (leave data on GPU)

Performance profile of original COSMO-2 model

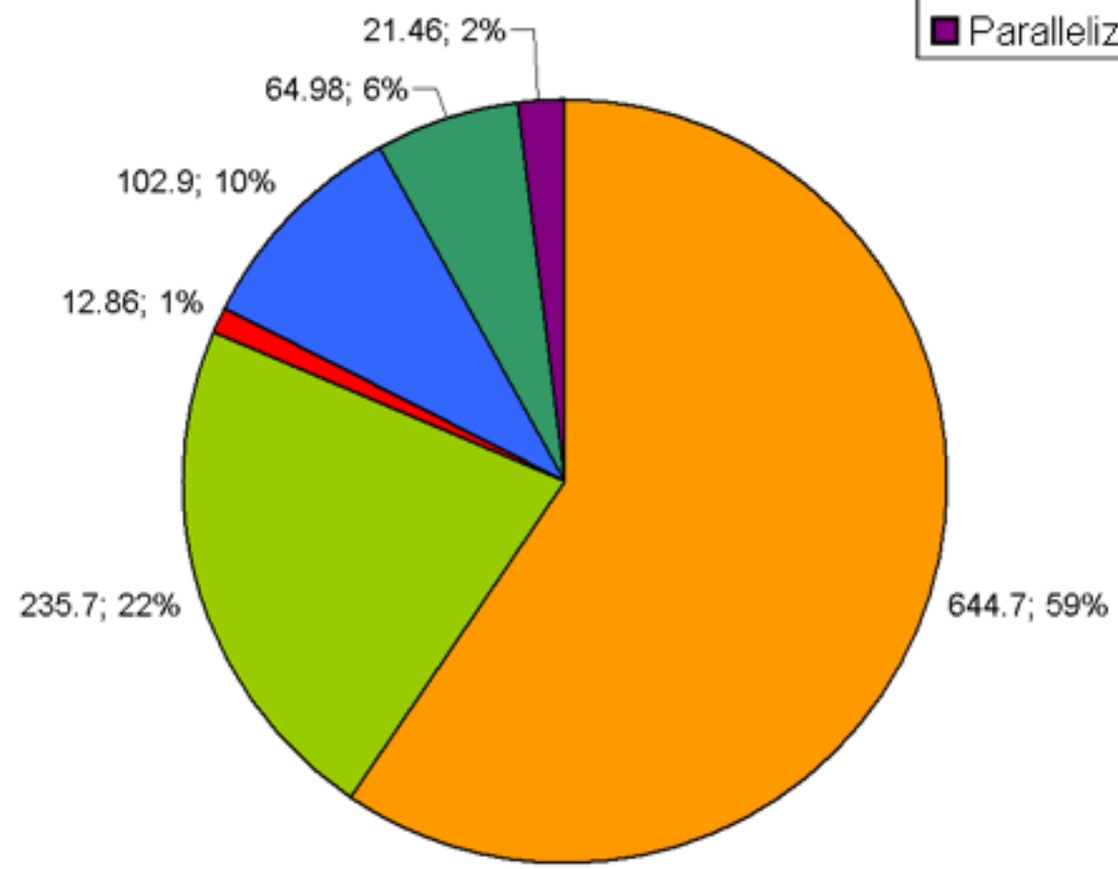
Runtime based 2 km production model of MeteoSwiss



% Code Lines (F90)



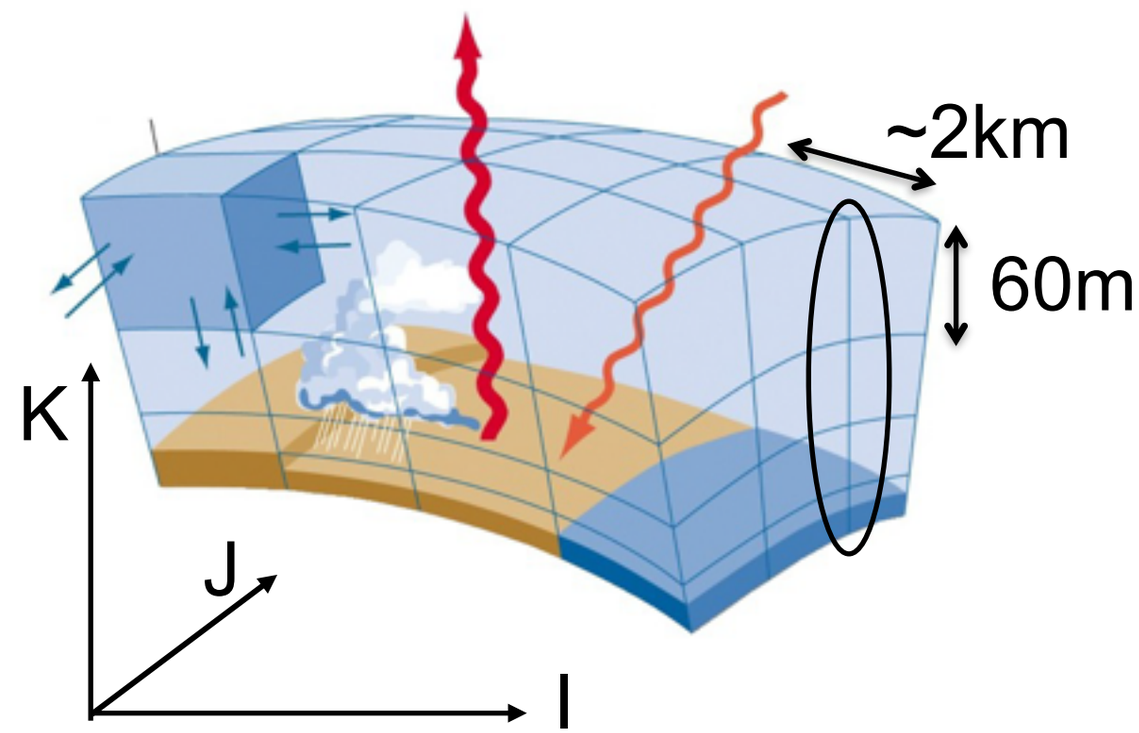
% Runtime



Original code (with OpenACC for GPU) **Rewrite in C++ (with CUDA backend for GPU)**

Insights into model, methods, and algorithms used in simulations with COSMO

- Solve PDE on structured grid where variables are velocity, temperature, pressure, humidity, etc.
- Explicit solve in horizontally (I,J) using finite differences
- Implicit solve in vertical direction (K), i.e. tridiagonal solve in every column

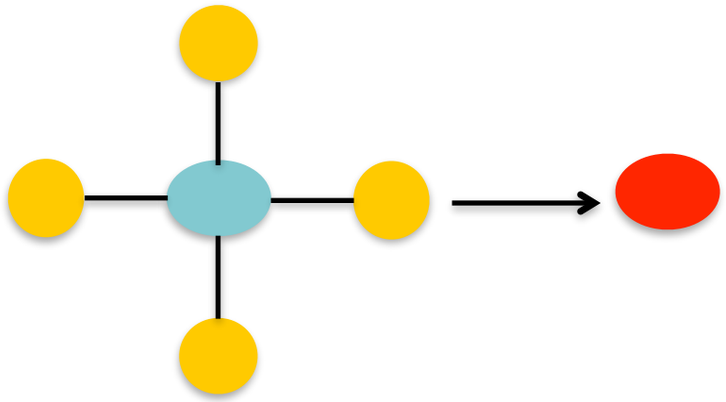


Implicit solve over columns allows use of longer time steps (now determined by 2km lateral grid rather than 60m shorts spacing)

Stencil computation

Stencils update array elements with a fixed pattern

```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```



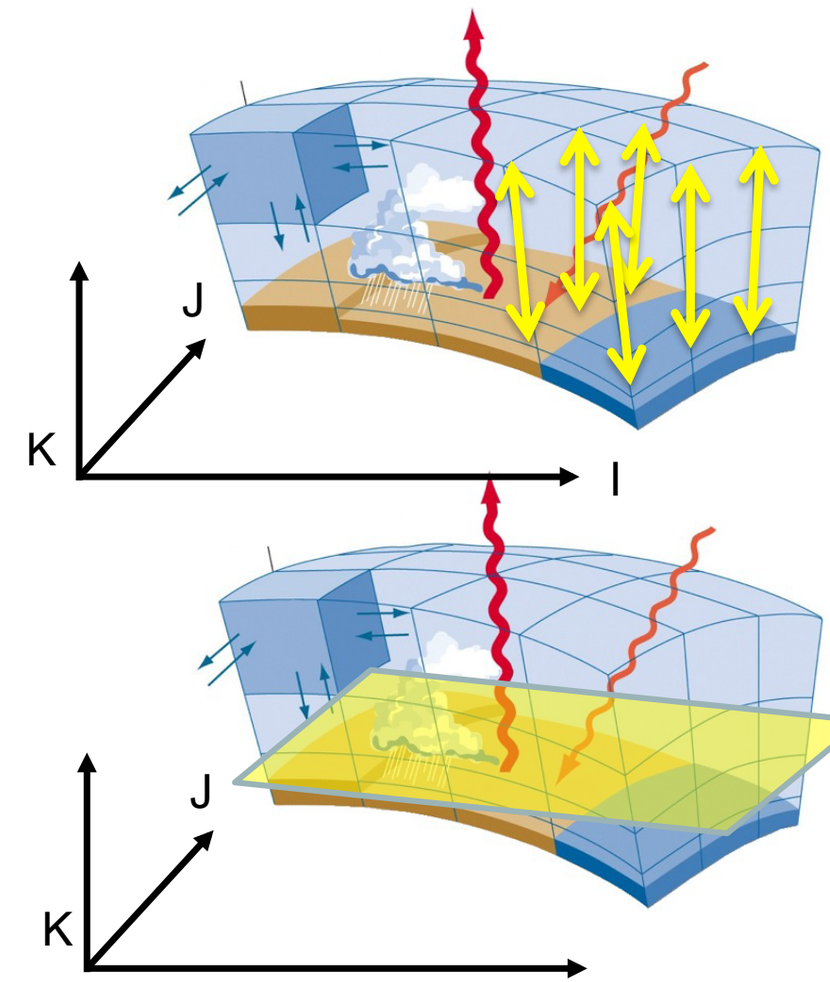
e.g. Laplacian: 5 flops per 6 load/stores ~ 0.1 flops/Byte

Kepler K20x GPU has 1310 Gflops / 200 GB/sec. ~ 6.6 flops/Byte

Stencil computations are memory bandwidth bound

To make matters more complicated

Tridiagonal solve in vertical
- ➔loop carried dependencies in K
- ➔focus on K access



Horizontal stencil computations
- ➔focus on IJ access
- ➔no loop carried dependencies

The two main motifs used in COSMO have different data access patterns – optimal storage order may depend on architecture

Dynamics in COSMO

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

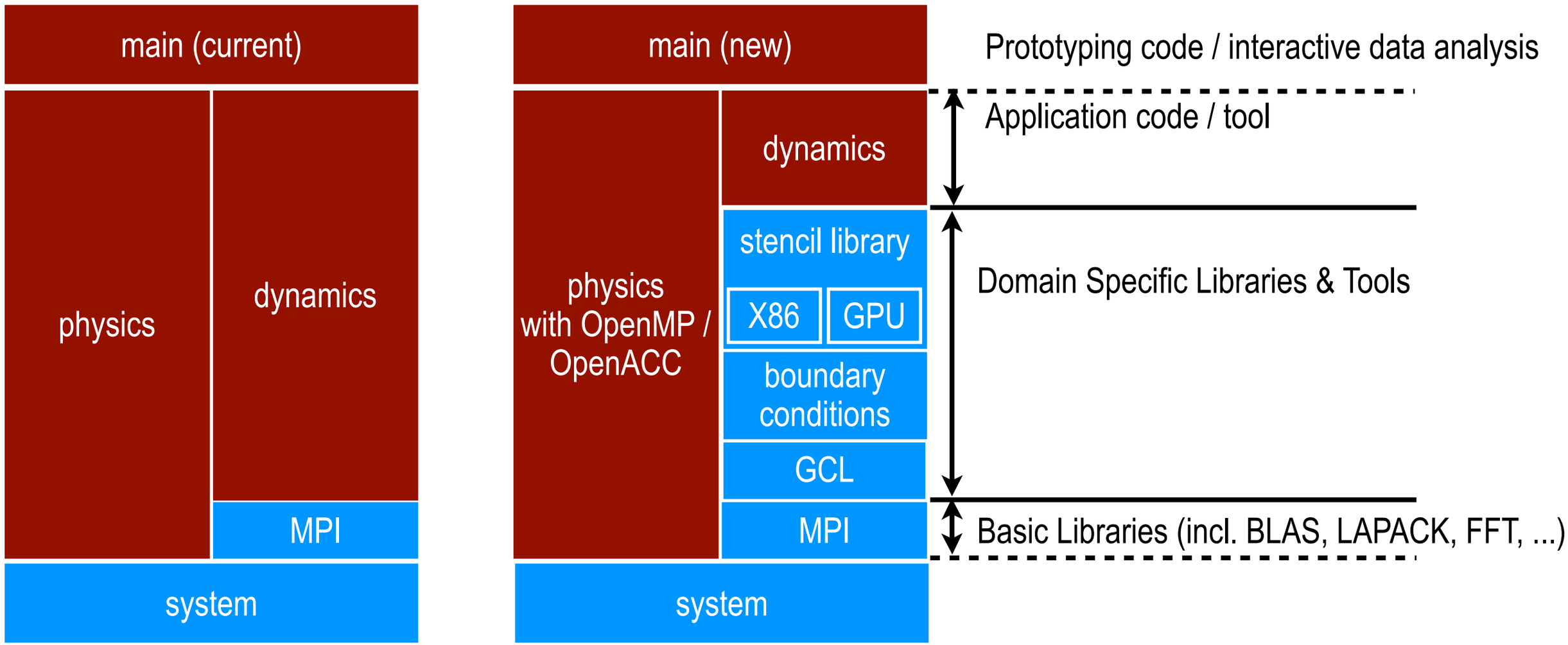
water adv.

Implement dynamics in terms of a stencil library

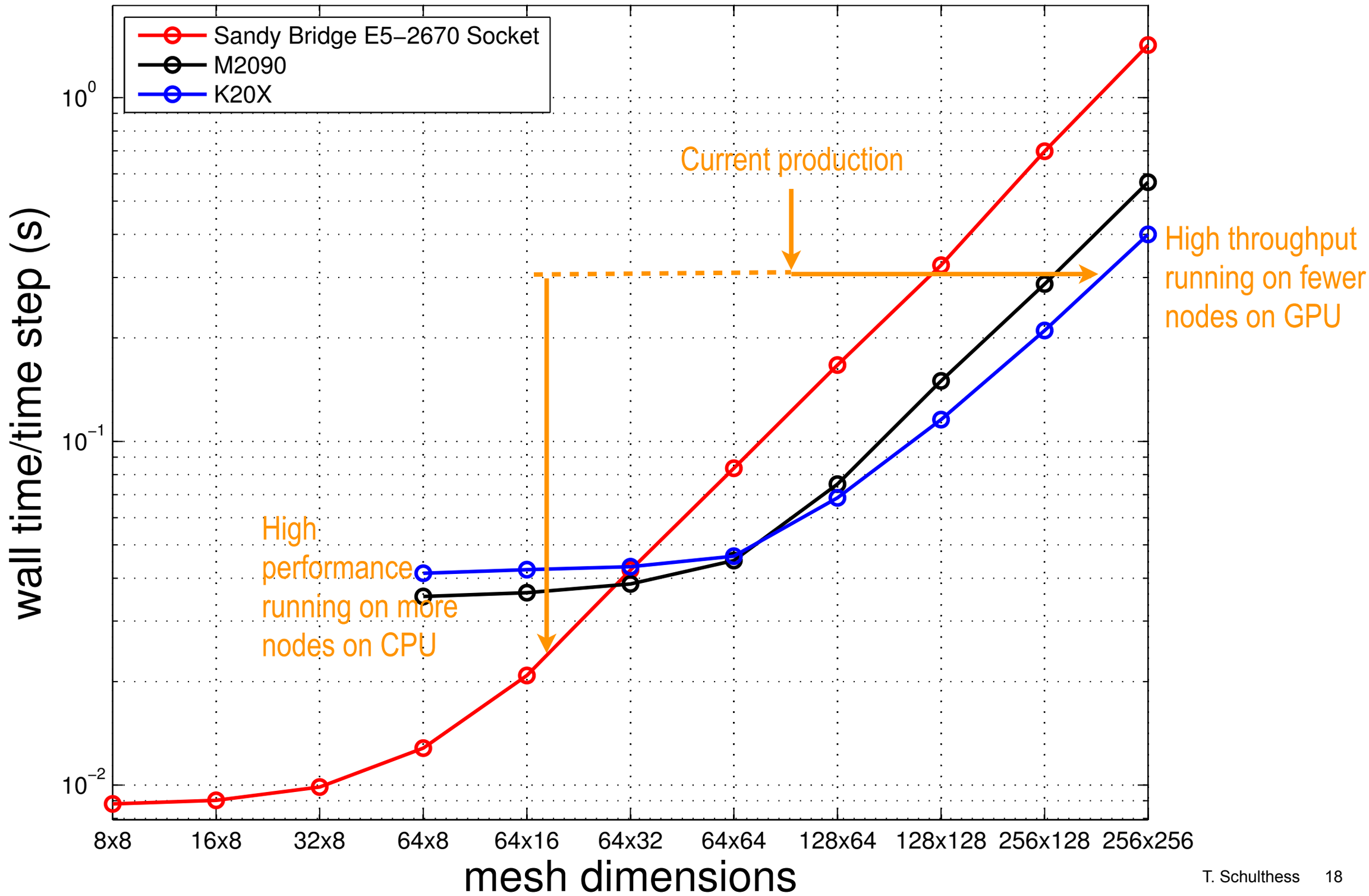
- Generic library using C++ and template meta-programming
  - currently support 3D structured grid
  - parallel over horizontal IJ-plane (sequential in K for tridiagonal solve)
  - multi-node support with halo exchange (Generic Communication Library)
- Abstract hardware platform (library runs optimally on CPUs and GPUs)
  - adapt loop order and storage layout to hardware platform
  - leverage software cashing
- Hide complex optimisation from users / programmer
- Since source code compiles to multiple platforms
  - current backends support X86 and CUDA

	X86 CPU	Tesla GPU
storage order (Fortran notation)	KIJ	IJK
parallelisation model	OpenMP	CUDA

Software architecture overview of current and new COSMO production code

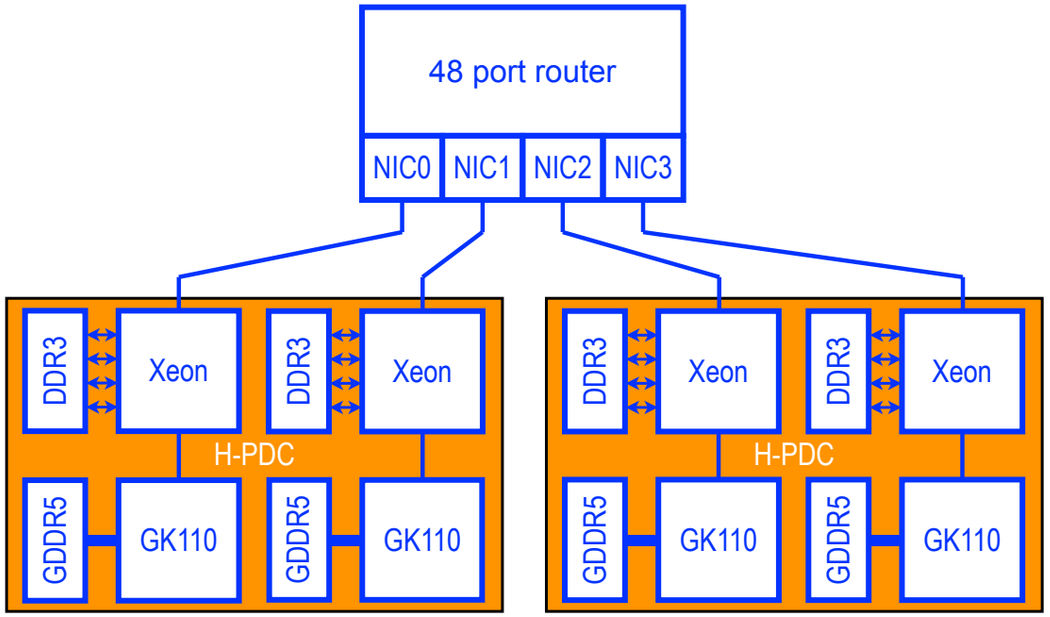


COSMO dynamical core running on CPU or GPU

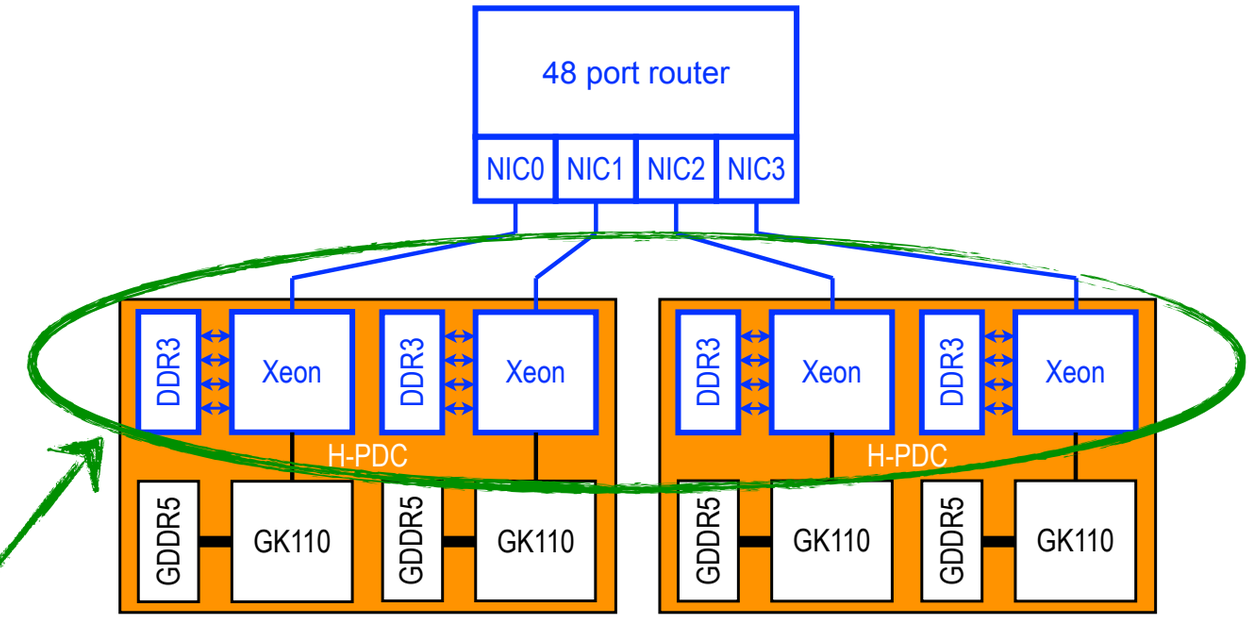


Three ways to look at hybrid CPU-GPU blades

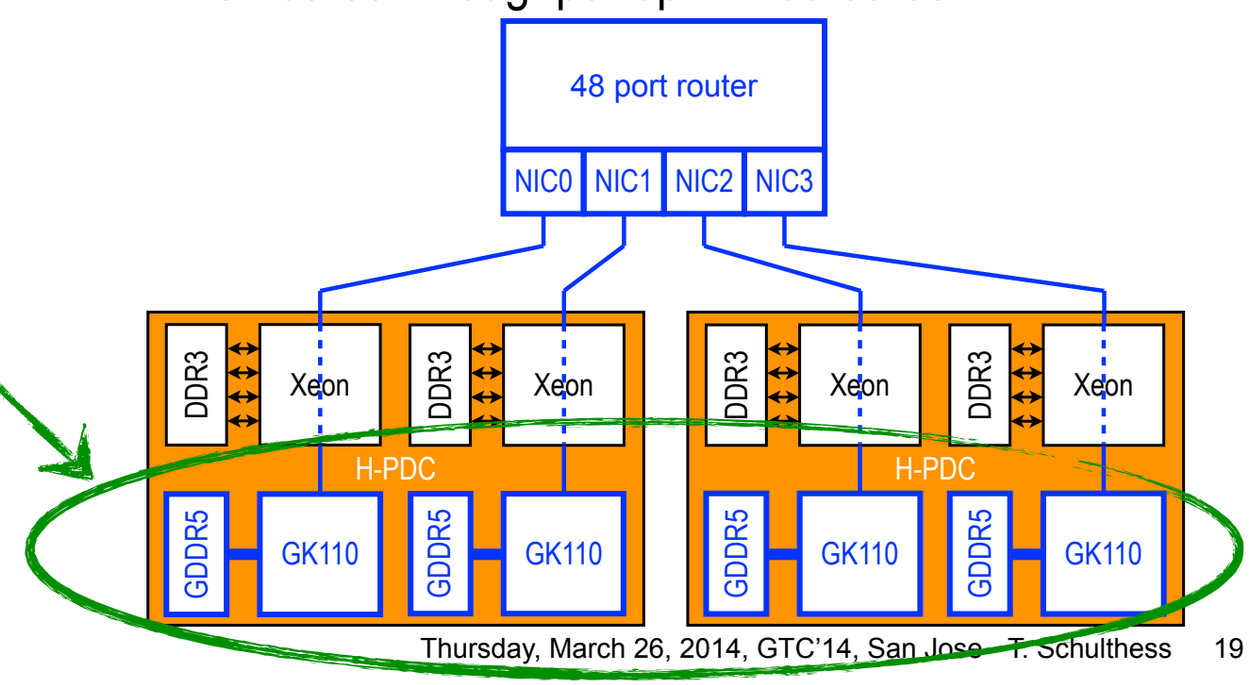
Standard: GPU is an accelerator



"Distributed latency optimized cores"



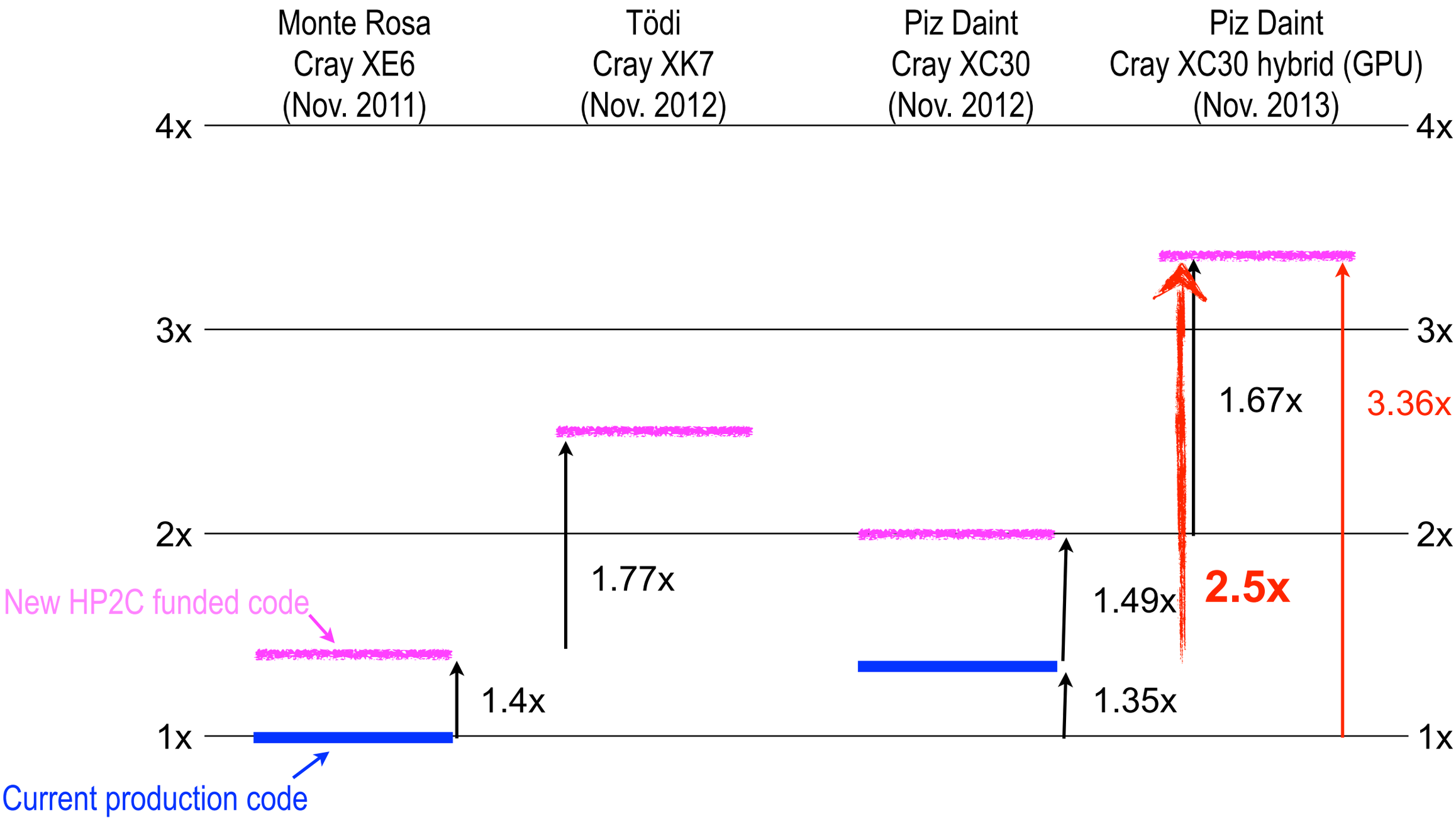
"Distributed throughput optimized cores"



Depending on workload, COSMO runs in either of these two modes:

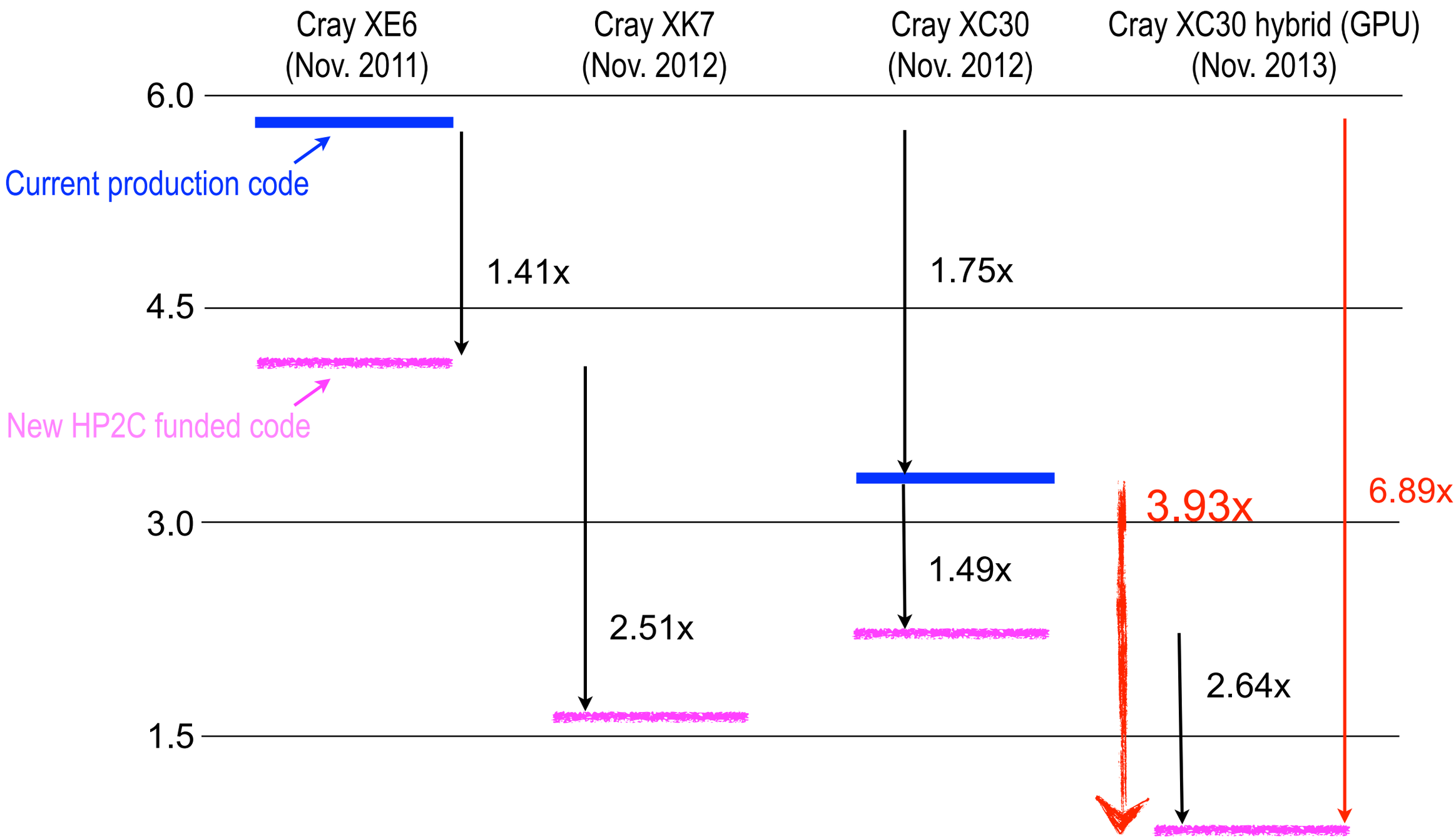
- **> domain decompose data over DDR3 or GDDR5 memory**
- **> move the computation close to data**
- **> exchange halos between CPUs or GPUs (with G-2-G communications)**

Speedup of the full COSMO-2 production problem (apples to apples with 33h forecast of Meteo Swiss)



all runs performed on same number of nodes

Energy to solution (kWh / ensemble member)



ETH zürich

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

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

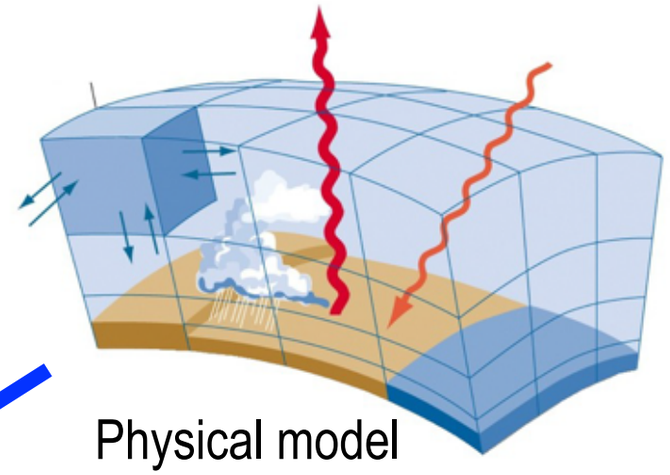
water

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

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

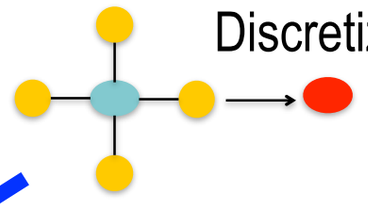
turbulence

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



Physical model

Mathematical description



Discretization / algorithm

Domain science (incl. applied mathematics)

```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Code / implementation

Code compilation

A given supercomputer

"Port" serial code to supercomputers

- > vectorize
- > parallelize
- > petascaling
- > exascaling
- > ...

Computer engineering (& computer science)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

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

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

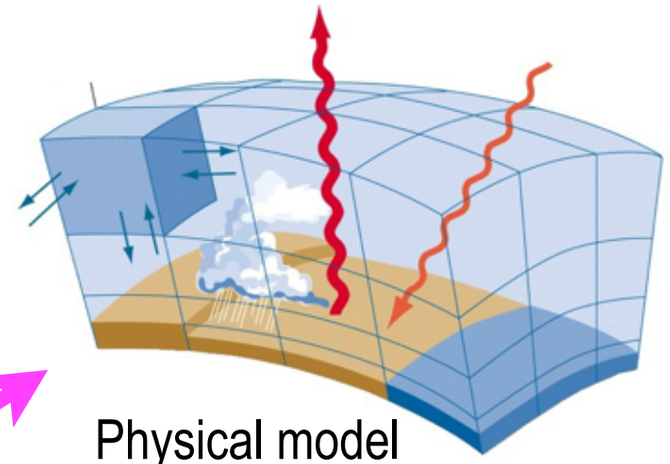
water

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

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

turbulence

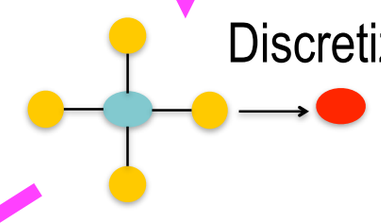
$$\frac{\partial e_t}{\partial t} = -\left\{\frac{1}{a\cos\varphi}\left(u\frac{\partial e_t}{\partial \lambda} + v\cos\varphi\frac{\partial e_t}{\partial \varphi}\right)\right\} - \dot{\zeta}\frac{\partial e_t}{\partial \zeta} + K_m^v\frac{g\rho_0}{\sqrt{\gamma}}\left\{\left(\frac{\partial u}{\partial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



Physical model

Mathematical description

Discretization / algorithm



Tools & Libraries

Domain science (incl. applied mathematics)

```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Code / implementation

Code compilation

Optimal algorithm
Auto tuning

Architectural options / design

Computer engineering (& computer science)

ETH zürich

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

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

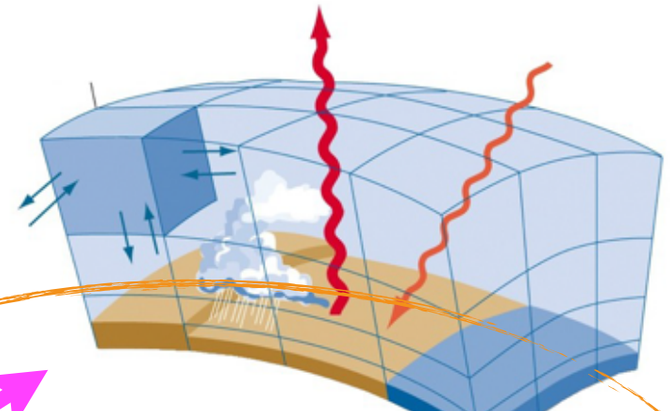
water

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

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

turbulence

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

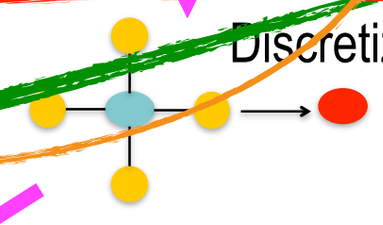


Physical model

Model development based on Python or equivalent dynamic language

Mathematical description

Discretization / algorithm



Domain science

```
lap(i,j,k) = -4.0 * data(i,j,k) +
    data(i+1,j,k) + data(i-1,j,k) +
    data(i,j+1,k) + data(i,j-1,k);
```

Code / implementation

Tools & Libraries

Optimal algorithm
Auto tuning

Code compilation

Computer engineering
(& computer science)

Architectural options / design

PASC co-design projects

Thank you!

Thank you to a great team!

- **Oliver Fuhrer** (Meteo Swiss)
- **Tobias Gysi** and **David Müller** (Supercomputing Systems)
- **Xavier Lapillone** and **Carlos Osuna** (C2SM @ ETH Zurich)
- **Will Saywer**, **Mauro Bianco**, **Ugo Vareto**, **Ben Cummings**, **Gille Fourestey** (CSCS)
- **Tim Schröder** and **Peter Messmer** (NVIDIA)
- **Ulli Schätler** and **Michael Baldauf** (Deutscher Wetterdienst)
- Many helpful discussions with **Christoph Schär** and **Torsten Höfler**