ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

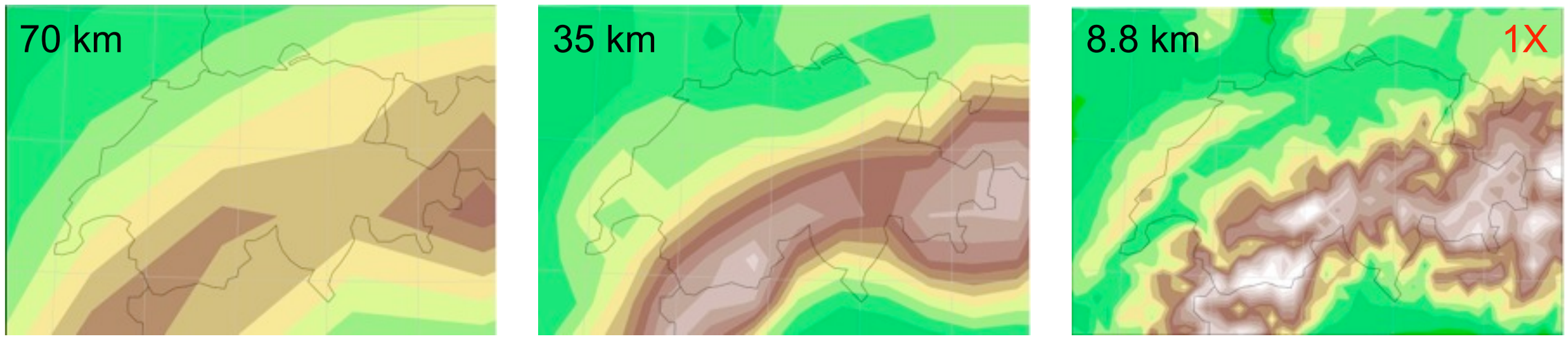
Hybrid CPU-GPU solutions for weather and cloud resolving climate simulations

Oliver Fuhrer,[1] Tobias Gysi,[2] Xavier Lapillonne,[3] Carlos Osuna,[3] Ben Cumming,[4] Mauro Bianco,[4] Ugo Vareto,[4] Will Sawyer,[4] Peter Messmer,[5] Tim Schröder,[5] and Thomas C. Schulthess[4]

with input from Jürg Schmidli,[6] Christoph Schär,[6] Isabelle Bey,[4] and Uli Schättler[7]

(1) Meteo Swiss, (2) SCS, (3) C2SM, (4) CSCS, (5) NVIDIA, (6) Inst. f. Atomospheric and Climate Science, ETH Zurich
(7) German Weather Service (DWD)

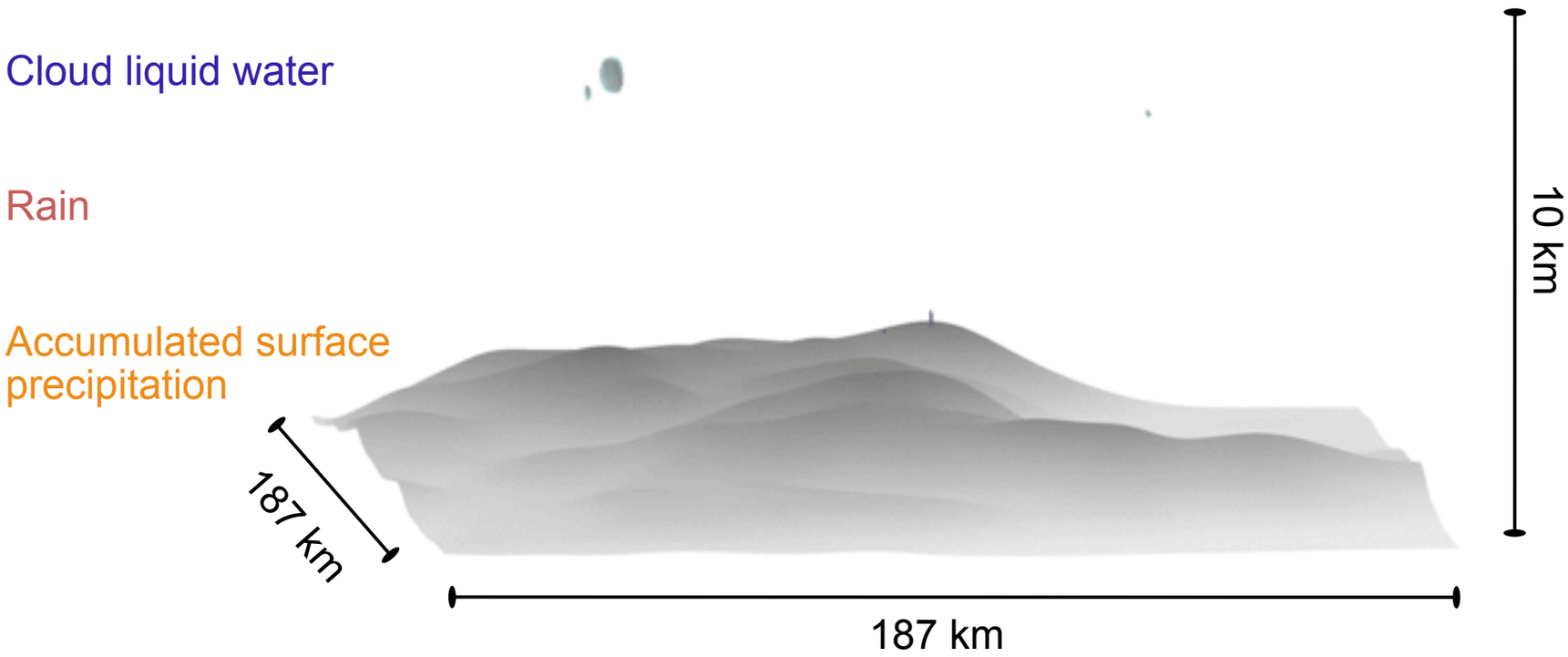
Why resolution is such an issue for Switzerland



Source: Oliver Fuhrer, MeteoSwiss

Cloud-resolving simulations

Breakthrough: Institute for Atmospheric and Climate Science Study at ETH Zürich (Prof. Schär) demonstrates cloud resolving models converge at 1-2km resolution



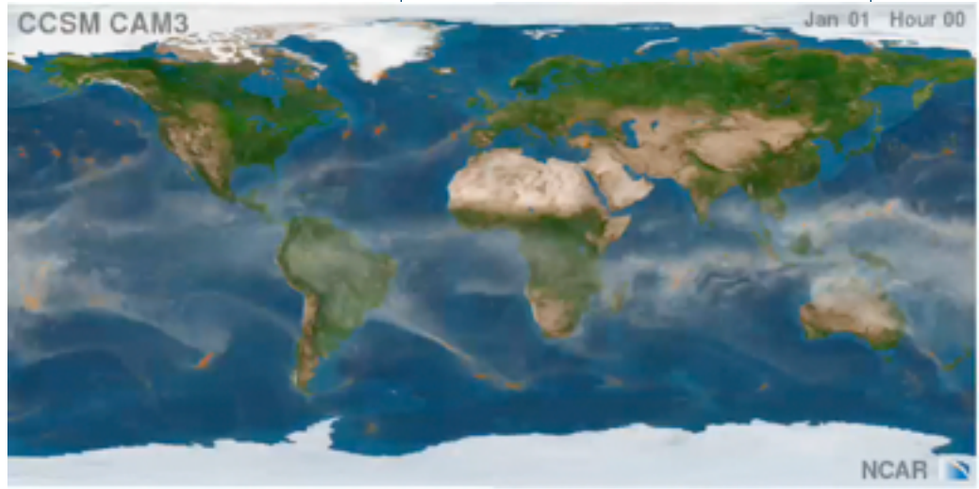
COSMO model setup: Δx=550 m, Δt=4 sec Plots generated using INSIGHT

Orographic convection – simulation: 11-18 local time, 11 July 2006 (Δt_plot=4 min)

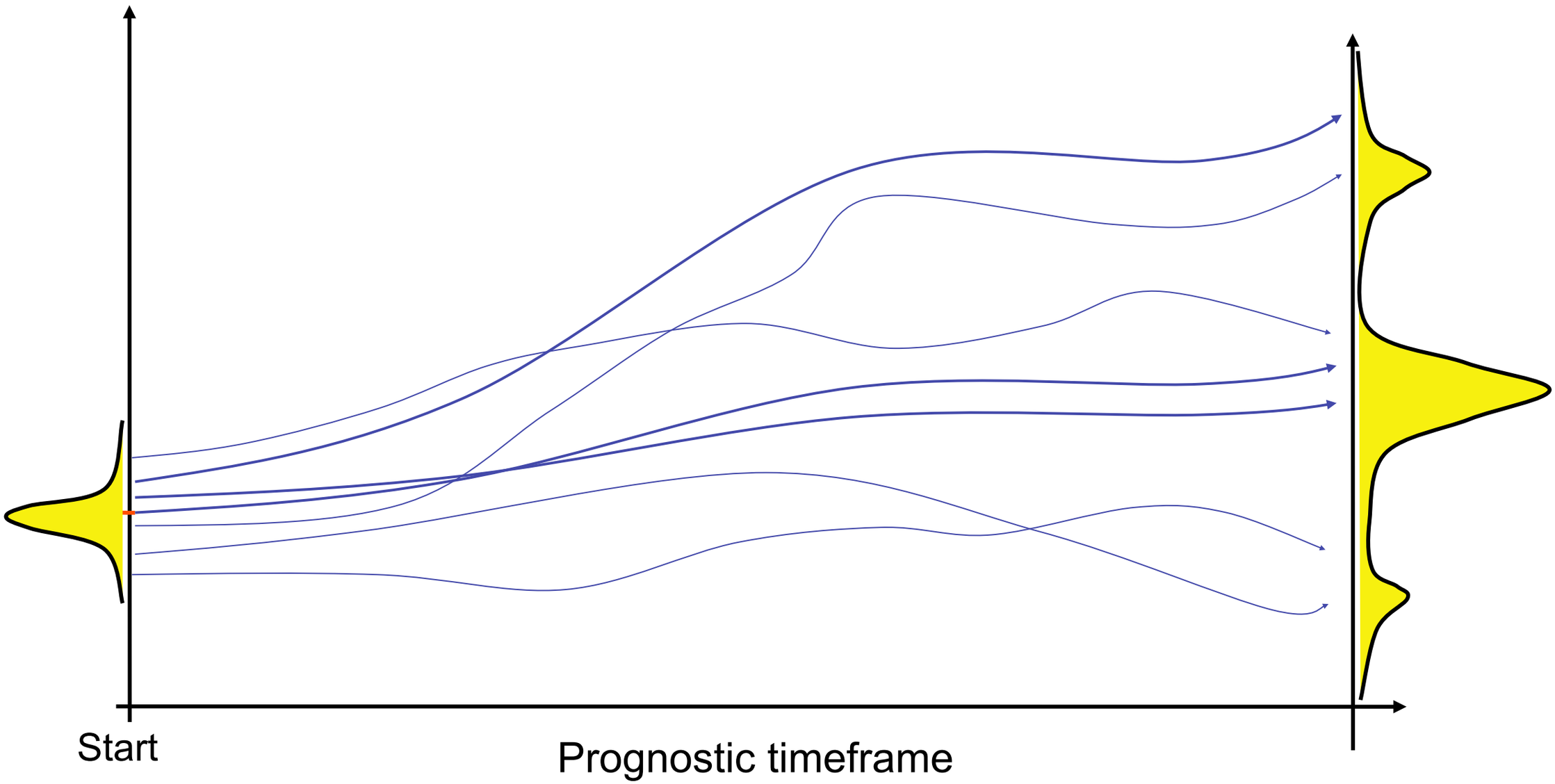
Source: Wolfgang Langhans, Institute for Atmospheric and Climate Science, ETH Zurich

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

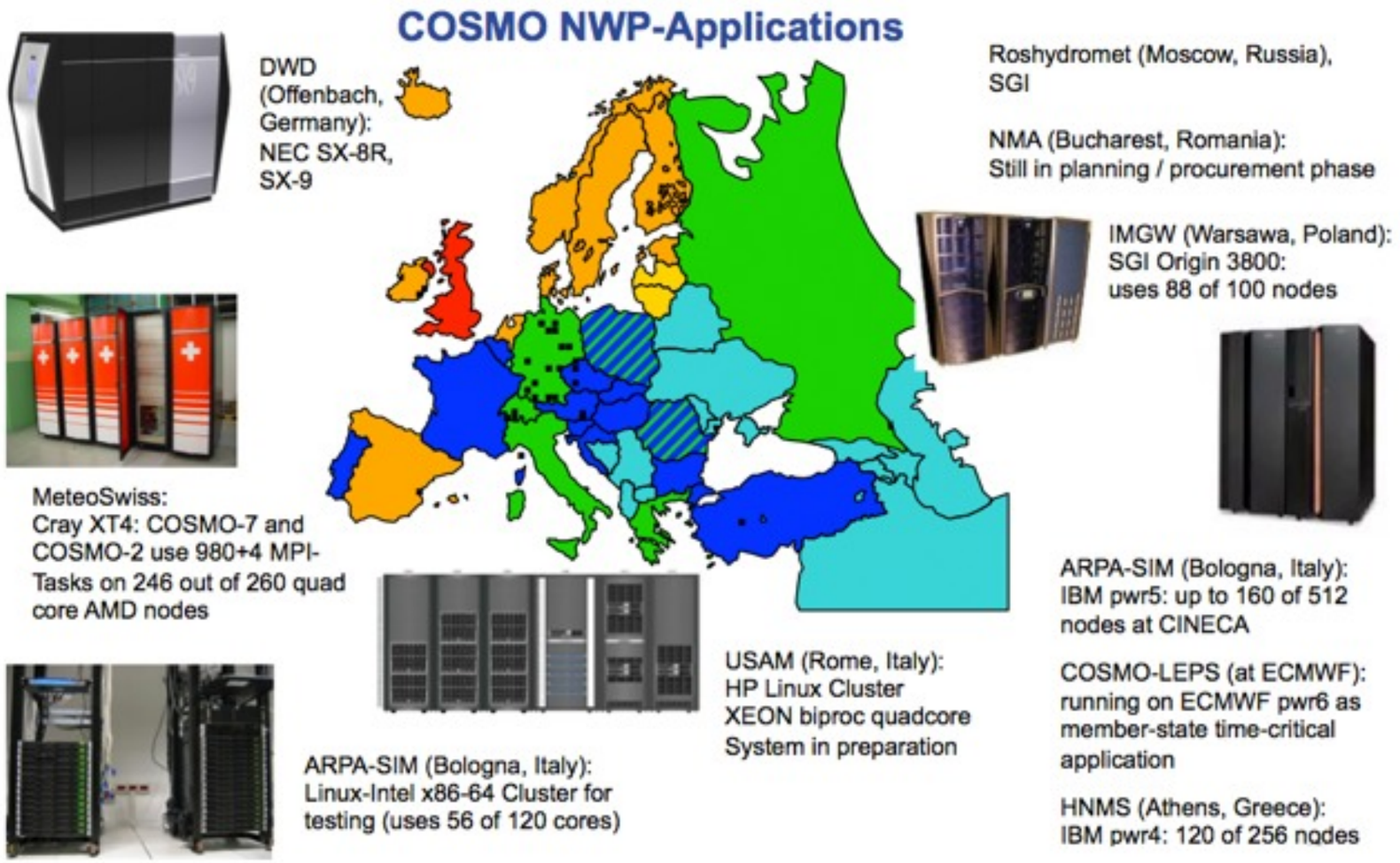
Source: Oliver Fuhrer, MeteoSwiss

WE NEED SIMULATIONS AT 1-2 KM

RESOLUTION AND THE ABILITY TO RUN ENSEMBLES AT THIS RESOLUTION

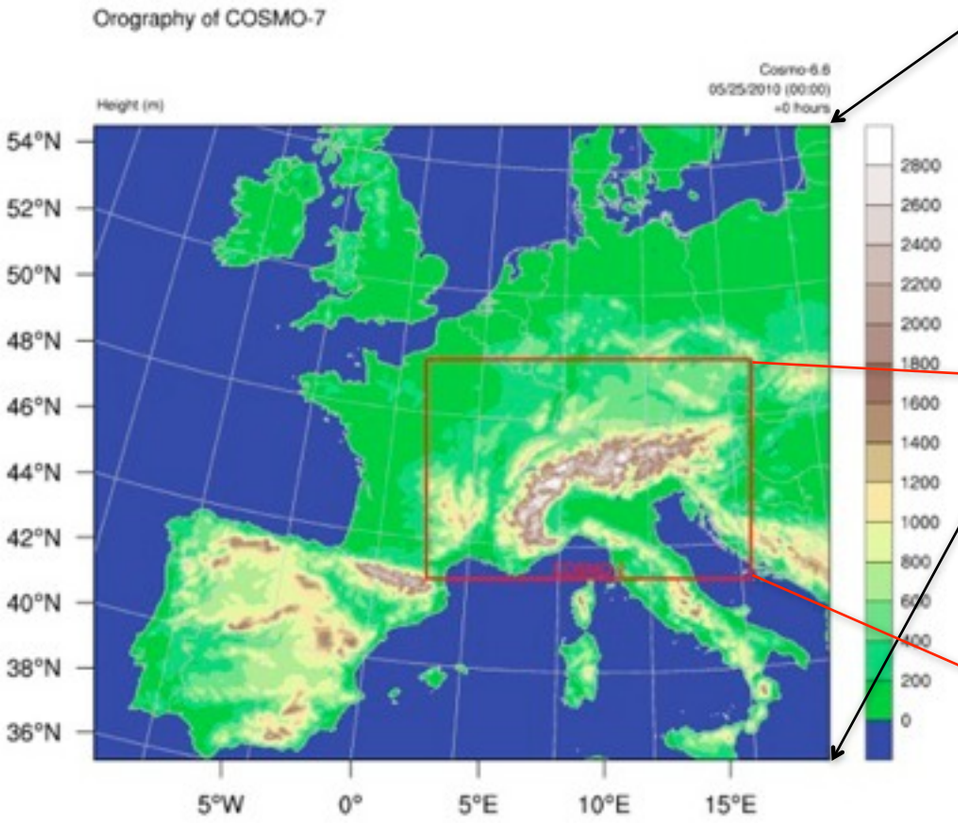
What is COSMO?

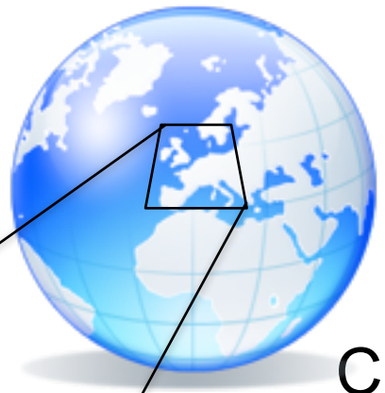
- Consortium for Small-Scale MOdeling
- Limited-area climate model (see http://www.cosmo-model.org)
- Used by 7 weather services as well as ~50 universities / research institutes



COSMO in production for Swiss weather prediction

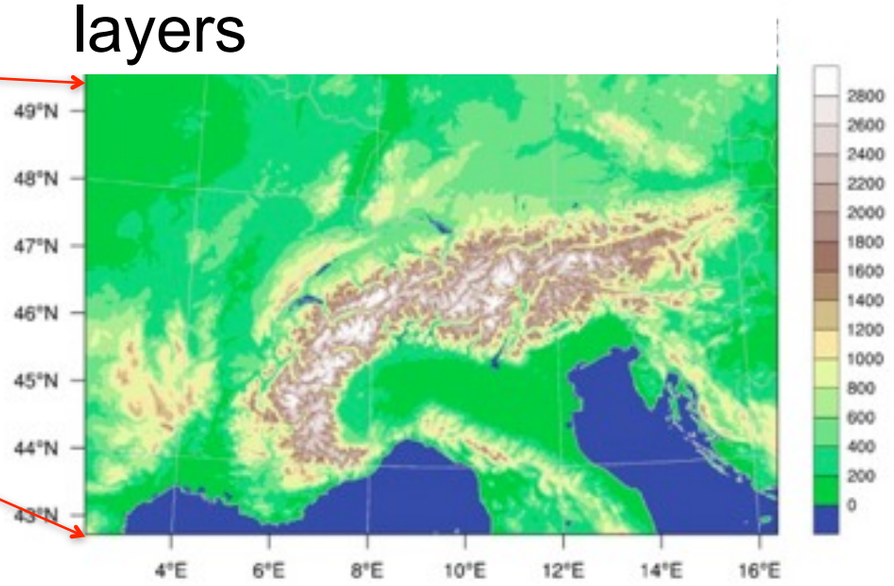
COSMO-7
3x per day 72h forecast
6.6 km lateral grid, 60 layers





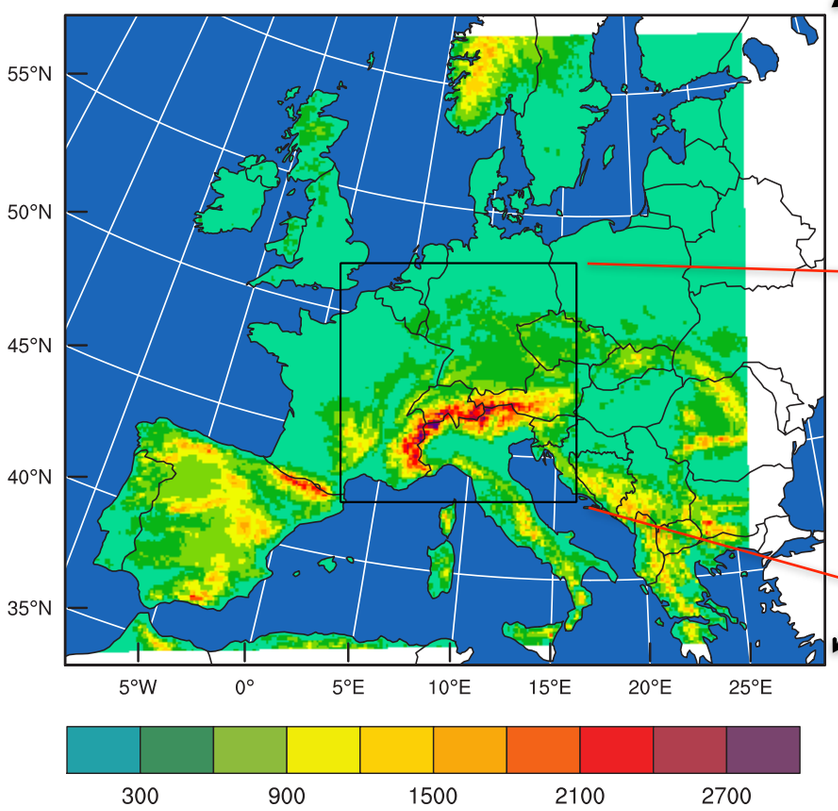
ECMWF
2x per day
16 km lateral grid, 91 layers

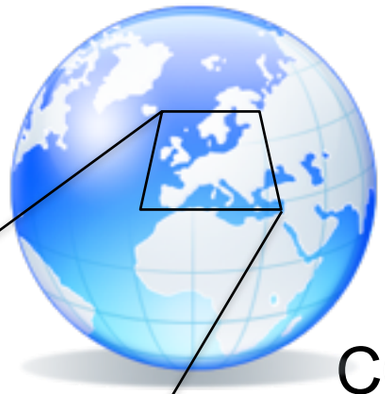
COSMO-2
8x per day 24h forecast
2.2 km lateral grid, 60 layers



COSMO-CLM in production for cloud resolving climate models

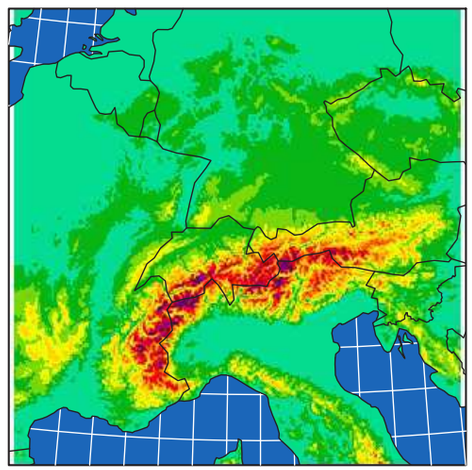
COSMO-CLM-12
12 km lateral grid, 60 layers
(260x228x60)





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

COSMO-CLM-2
2.2 km lateral grid, 60 layers
(500x500x60)



Simulating 10 years

Configuration is similar to that of COSMO-2 used in numerical weather prediction by Meteo Swiss

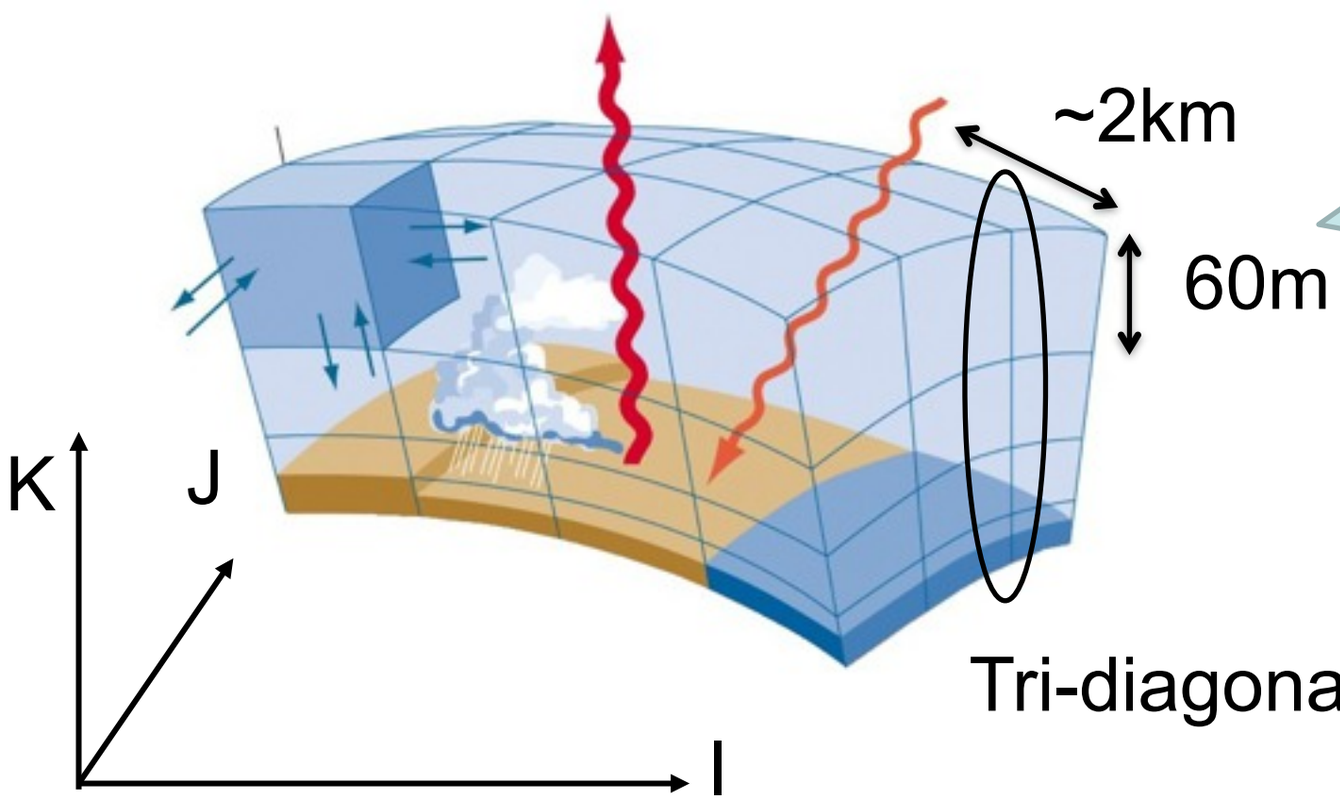
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

CAN WE ACCELERATE THESE SIMULATION BY 10X AND REDUCE THE RESOURCES USED PER SIMULATION FOR ENSEMBLE RUNS?

Insight into model/methods/algorithms used in COSMO

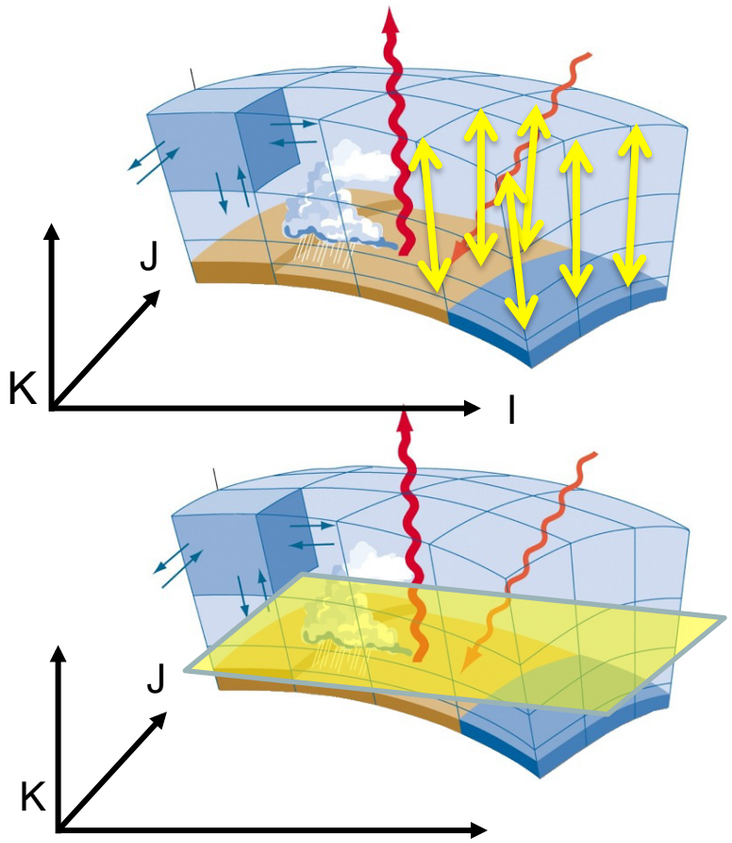
- PDE on structured grid (variables: velocity, temperature, pressure, humidity, etc.)
- Explicit solve horizontally (I, J) using finite difference stencils
- Implicit solve in vertical direction (K) with tri-diagonal solve in every column (applying Thomas algorithm in parallel – can be expressed as stencil)



Due to implicit solves in the vertical we can work with longer time steps (2km and not 60m grid size is relevant)

Hence the algorithmic motif in the dynamics are

- Tri-diagonal solve
  - vertical K-diretion
  - with loop carried dependencies in K

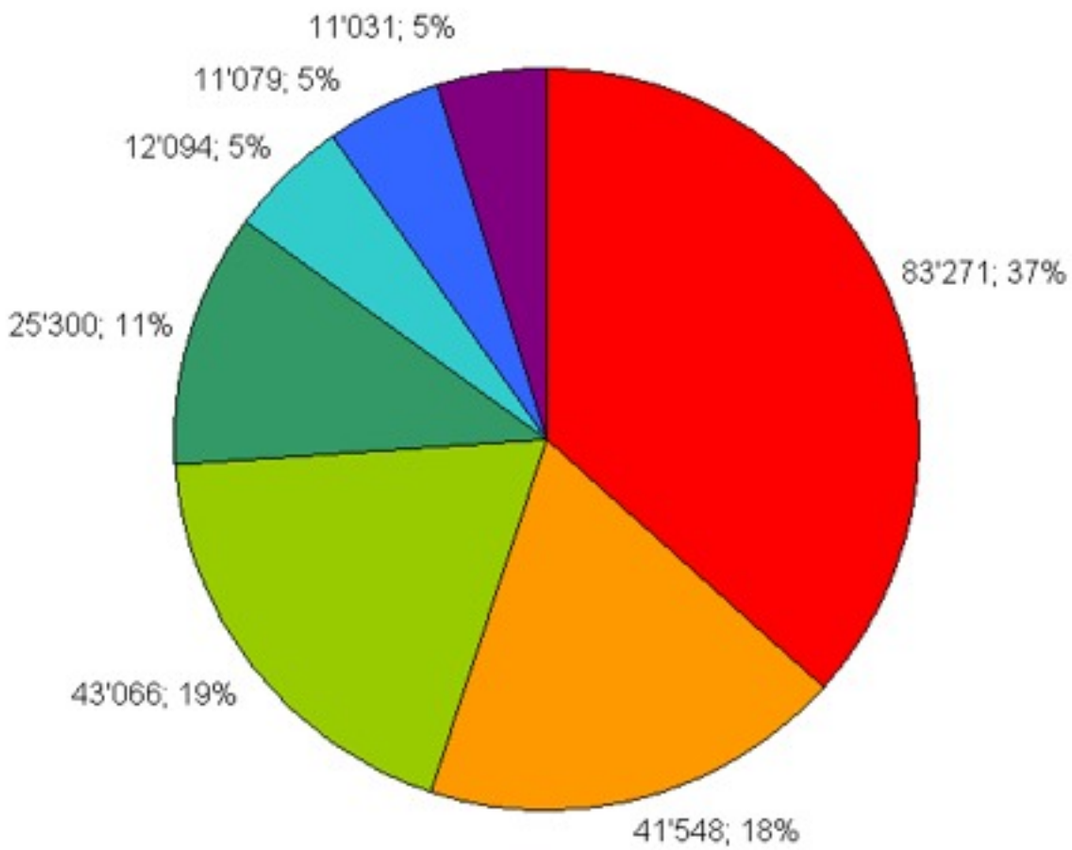


- Finite difference stencil computations
  - focus on horizontal IJ-plane access
  - no loop carried dependencies

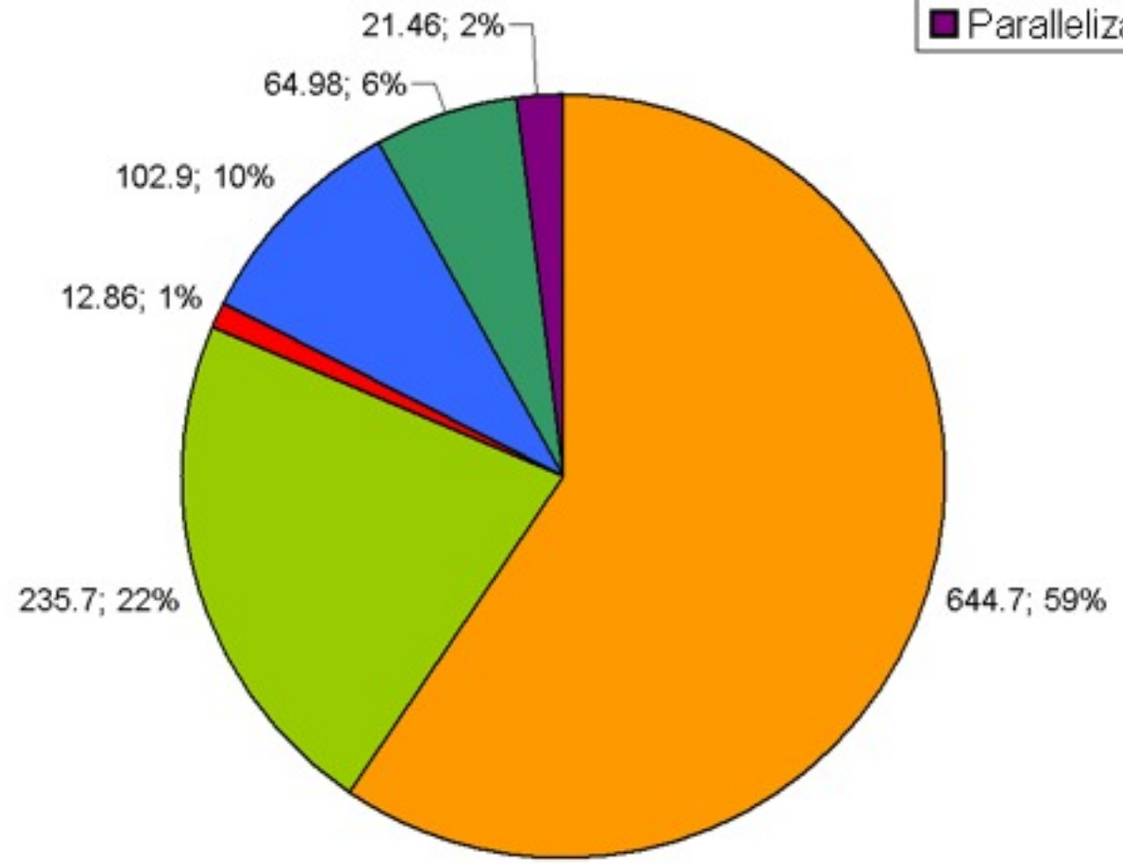
Performance profile of (original) COSMO-CCLM

Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90)



% Runtime



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

Strategies to improve performance

- Adapt code employing bandwidth saving strategies
  - computation on-the-fly
  - increase data locality
- Choose hardware with hight memory bandwidth (e.g. GPU)

	Peak Performance	Memory Bandwidth
Interlagos	147 Gflops	52 GB/s
Tesla 2090	665 Gflops	150 GB/s

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

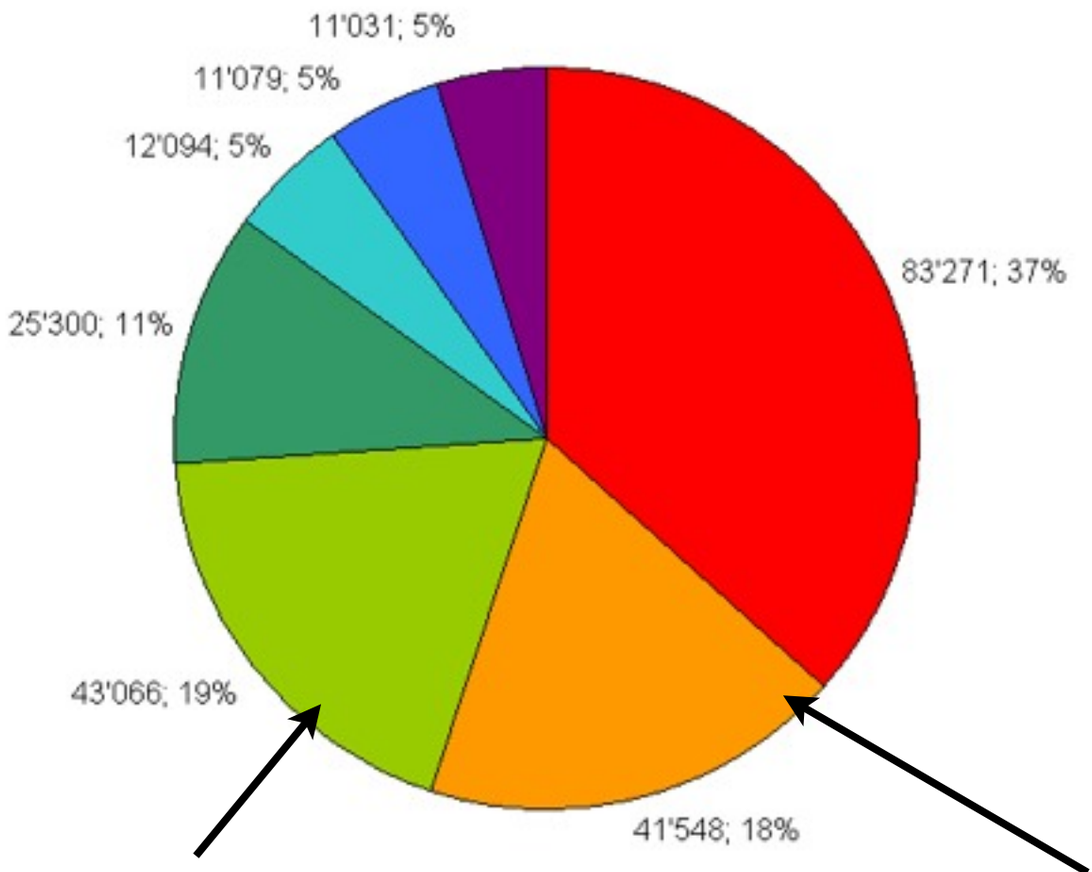
Machine	Interlagos	Fermi (2090)	GPU+transfer
Time	0.16 s	0.038 s	1.7 s
Speedup	**1.0 (REF)**	**4.2**	**0.1**

The simple lesson: leave data on the GPU!

Performance profile of (original) COSMO-CCLM

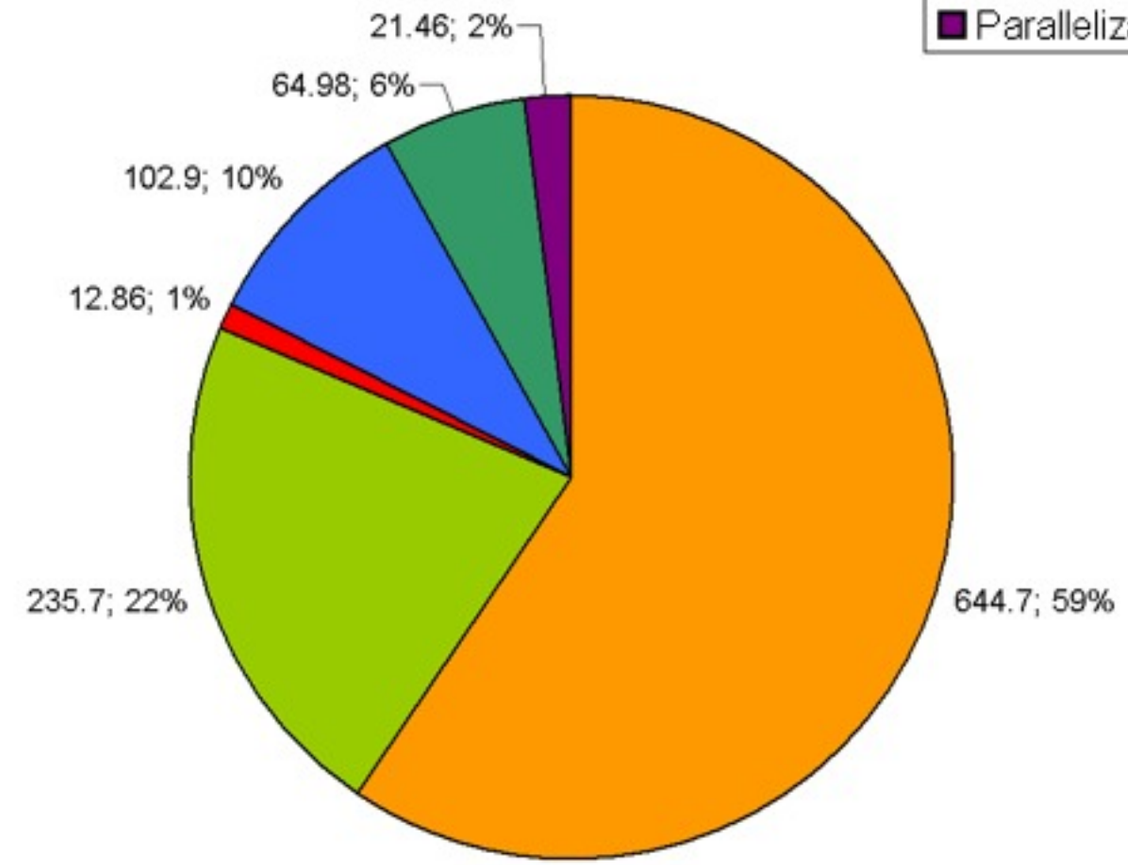
Runtime based 2 km production model of MeteoSwiss

% Code Lines (F90)



Original code (with OpenACC)

% Runtime



Rewrite in C++ (with CUDA backend)

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.

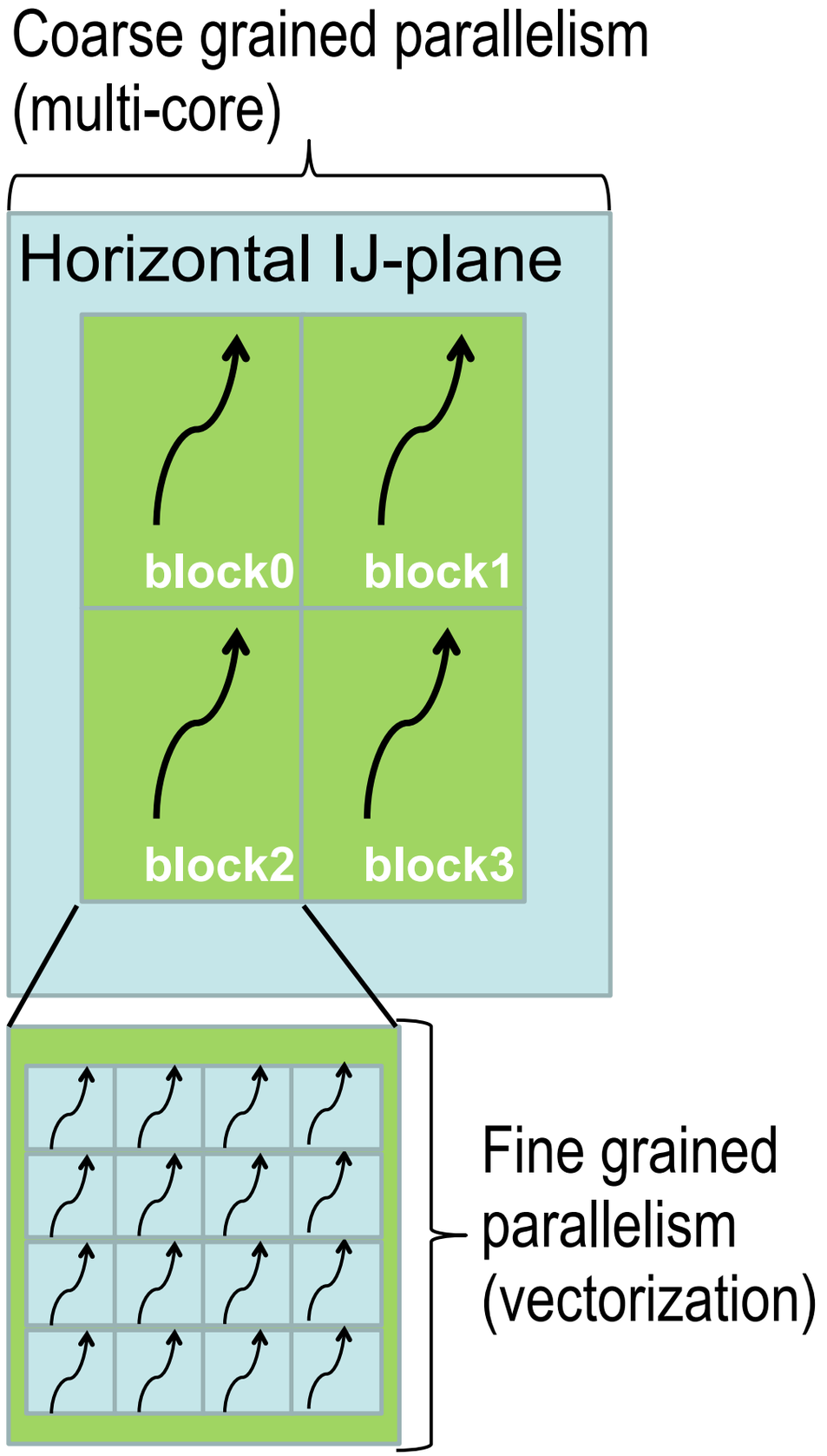
Stencil Library Ideas

- Implement a stencil library using C++ and template metaprogramming
  - 3D structured grid
  - Parallelization in horizontal IJ-plane (sequencial loop in K for tri-diagonal solves)
  - Multi-node support using explicit halo exchange (Generic Communication Library – not covered by presentation)
- Abstract the hardware platform (CPU/GPU/MIC)
  - Adapt loop order and storage layout to the platform
  - Leverage software caching
- Hide complex and “ugly” optimizations
  - Blocking

Stencil Library Parallelization

- Shared memory parallelization
  - Support for 2 levels of parallelism
- Coarse grained parallelism
  - Split domain into blocks
  - Distribute blocks to cores
  - No synchronization & consistency required
- Fine grained parallelism
  - Update block on a single core
  - Lightweight threads / vectors
  - Synchronization & consistency required

Similar to CUDA programming model
(should be a good match for other platforms as well)



Stencil Code Concepts

- Writing a stencil library is challenging
  - No big chunk of work suitable for a library call (unlike BLAS)
  - Countable but infinite number of interfaces – one interface per differential operator
- Resort to Domain Specific Embedded Language (DSEL) with C++ template meta programing
- A stencil definition has two parts
  - Loop-logic defining the stencil application domain and order
  - Update-function defining the update formula

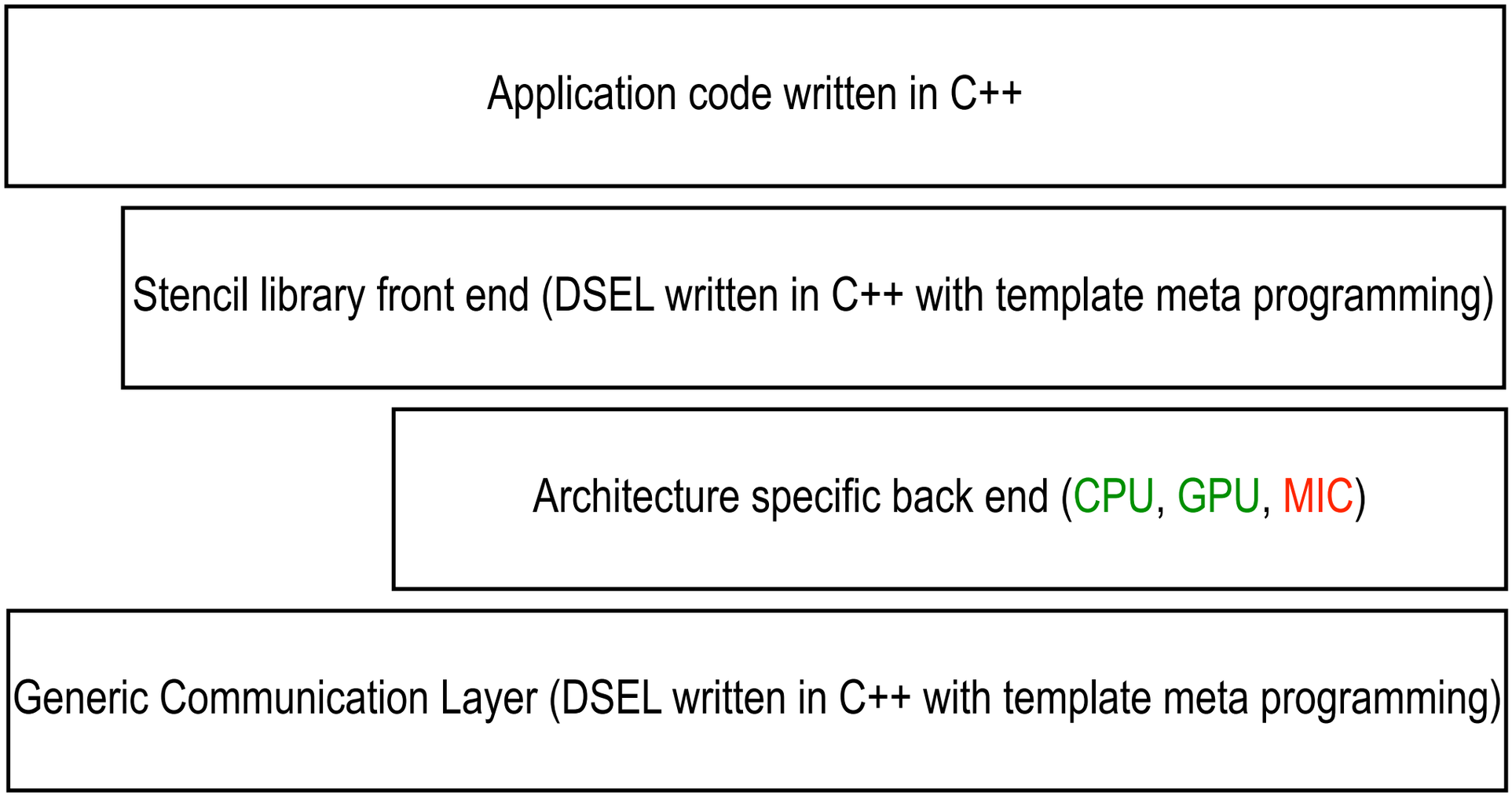
```
DO k = 1, ke
  DO j = jstart, jend
    DO i = istart, iend
      lap(i,j,k) = data(i+1,j,k) + data(i-1,j,k) + data(i,j+1,k) + data(i,j-1,k)
                   - 4.0 * data(i,j,k)
    ENDDO
  ENDDO
ENDDO
```

Stencil Library for COSMO Dynamical Core

- Library distinguished loop-logic and update functions
- Loop logic is defined using a domain specific language
  - Abstract parallelization / execution order of the update function
- Single source code compiles to multiple platforms
  - Currently, efficient back-ends are implemented for CPU and GPU

	CPU	GPU
Storage Order (Fortran notation)	KIJ	IJK
Parallelization	OpenMP	CUDA

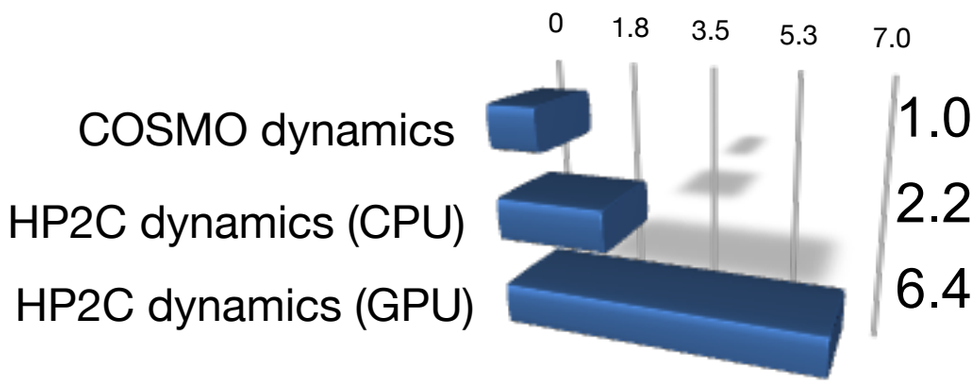
Software structure for new COSMO DyCore



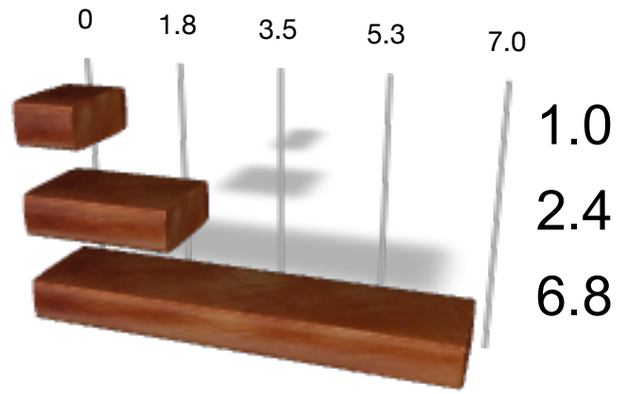
Application performance of COSMO dynamical core (DyCore)

- The CPU backend is 2x-2.9x faster than standard COSMO DyCore
  - Note that we use a different storage layout in new code
  - 2.9x applied to smaller problem sizes, i.e. HPC mode (see later slide)
- The GPU backend is 2.8-4x faster than the CPU backend
- Speedup new DyCore & GPU vs. standard DyCore & CPU = 6x-7x

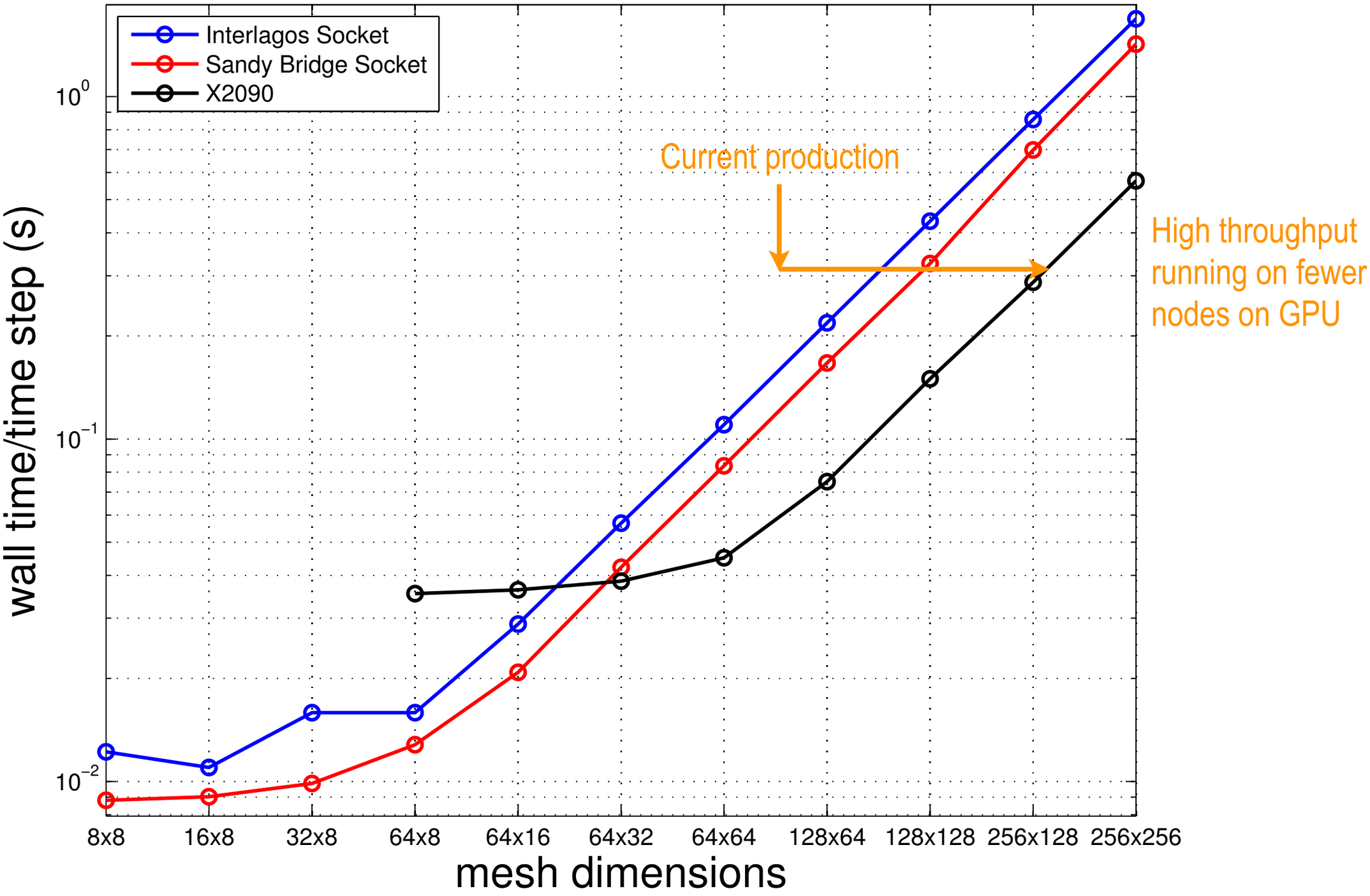
Interlagos vs. Fermi (M2090)



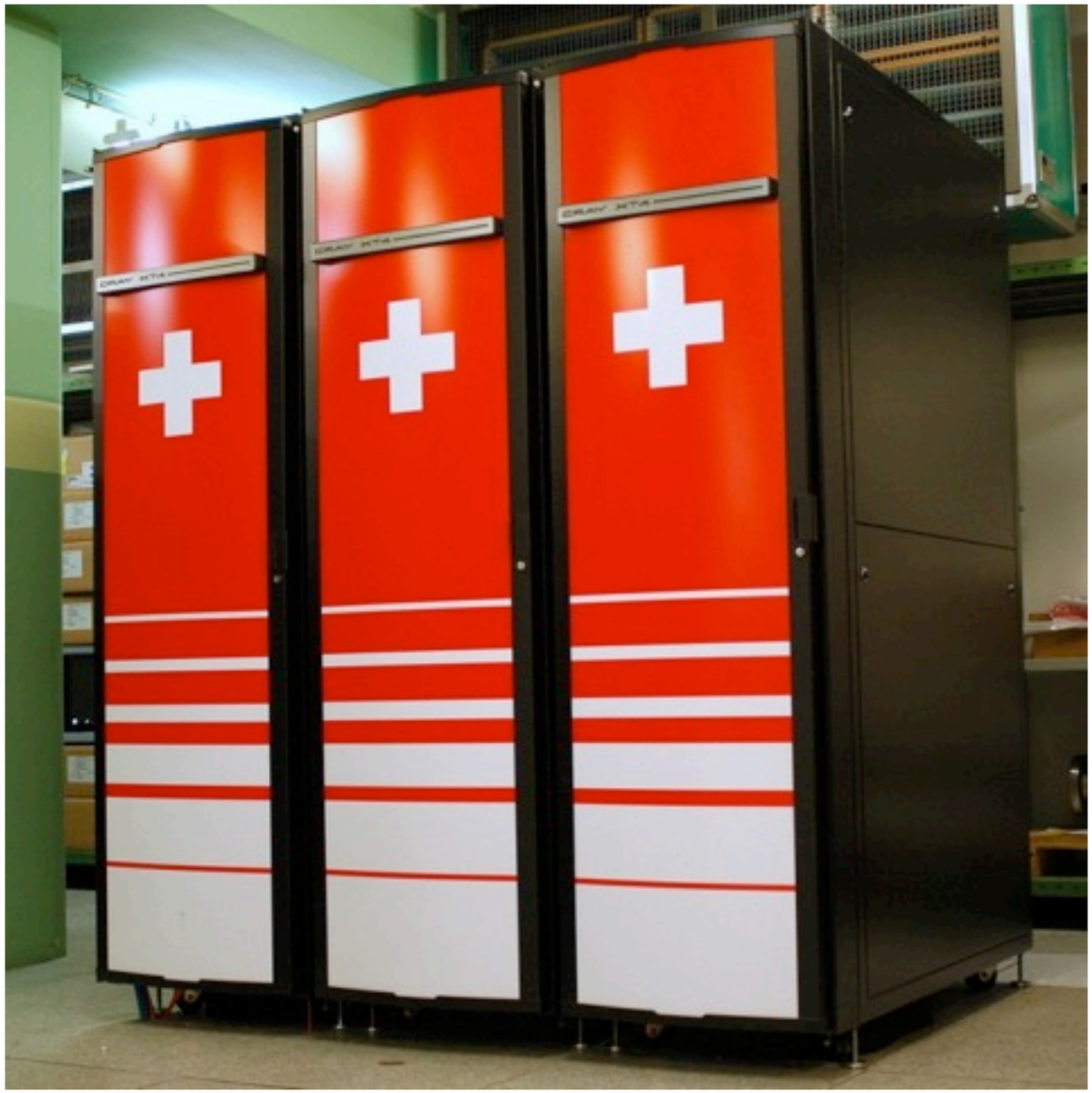
SandyBridge vs. Kepler



Ideal workloads for CPU and GPU (based on performance of dynamical core)

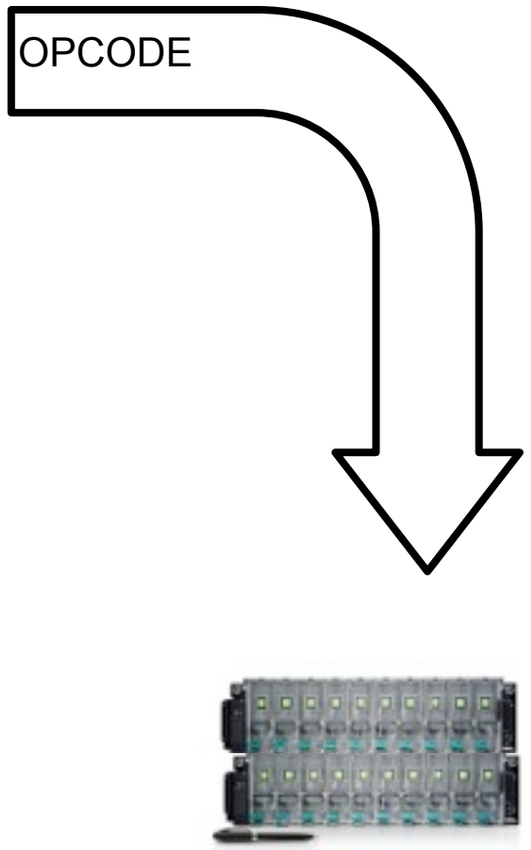


OPCODE project: can we run entire MeteoCH production suit on a node with ~8-16 GPUs?



Cray XT4 (Production Machine at CSCS)

246 AMD Opteron Barcelona

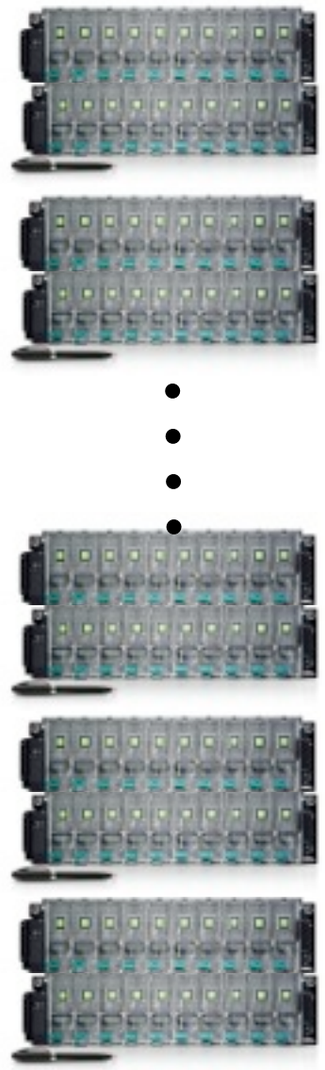


Server

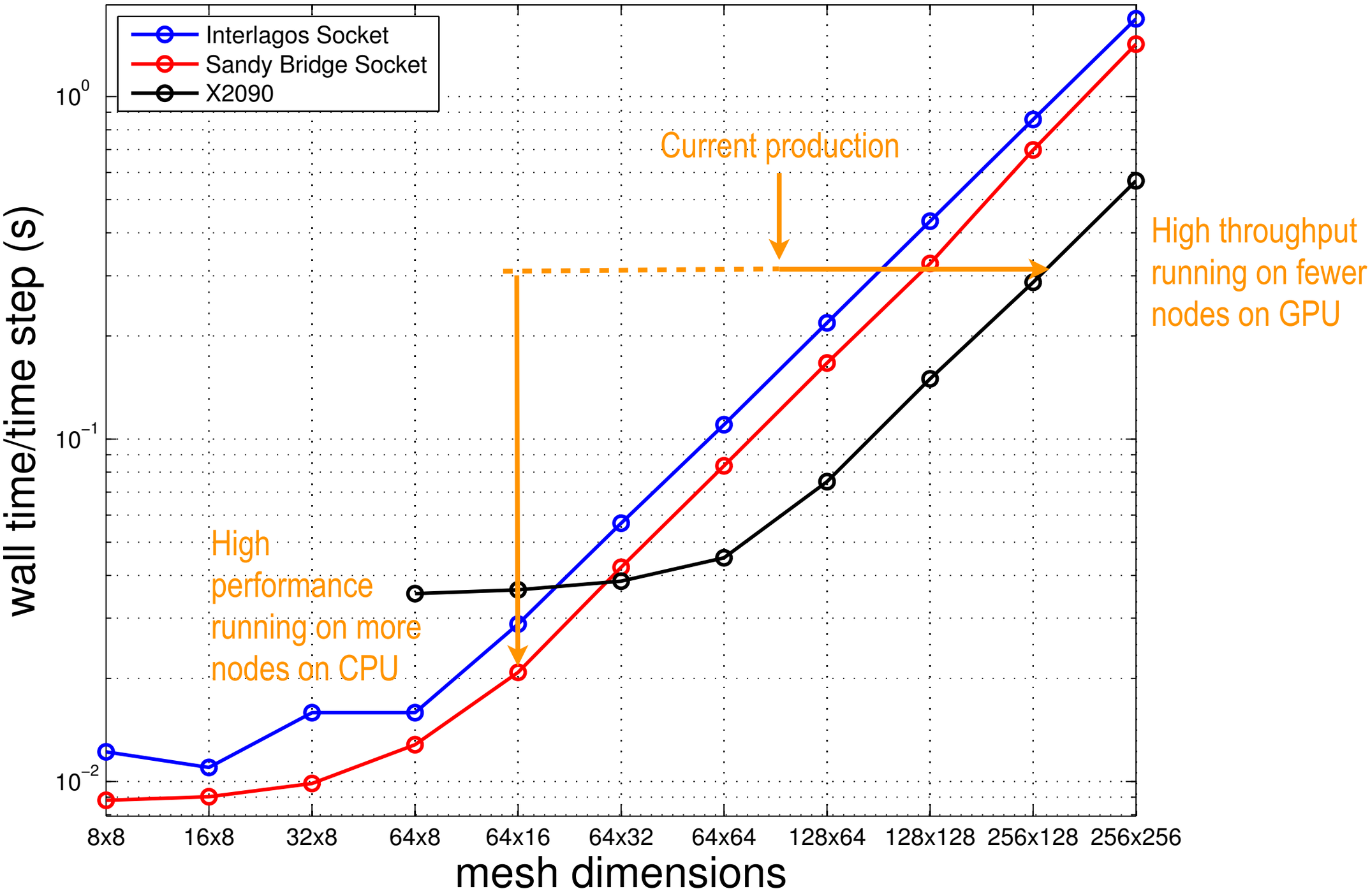
O(20) GPUs

Many such Server

for ensemble runs



Ideal workloads for CPU and GPU (presently based on performance of DyCore only)



THANK YOU!